Machine learning in SDN networks for secure industrial cyber physical systems: a case of detecting link flooding attack

Priyanshi Deliwala

School of Engineering and Applied Science,
Ahmedabad University, India
Email: priyanshi.d1@ahduni.edu.in

Rutvij H. Jhaveri*

Department of Computer Science and Engineering,
School of Technology, India
Email: Rutvij.Jhaveri@sot.pdpu.ac.in
*Corresponding author

Sagar Ramani

Gujarat Technology University,
Ahmedabad, India
Email: sagarramani@gmail.com

Abstract: Software-defined networking (SDN) is an emerging network architecture which has potential to solve real-time challenges of industrial cyber physical systems (ICPSs). However, it also opens a gateway to different network attacks such as link flooding attack (LFA). This paper illustrates how SDN's control layer is endangered to LFA. While the proposed approach (Iwendi et al., 2020a) can appear suitable on the surface, some weaknesses and anomalies are discovered when deliberated deeper. In the current paper, we point out these anomalies and the limitations of those anomalies by applying two machine learning algorithms, namely ANN-MLP and random forest to correctly classify the virulent traffic during congestion. We carry out experiments on Mininet network emulator and use the WEKA tool to assess the metrics by utilising the available datasets. The results show that a random forest can accurately detect virulent traffic and efficiently terminate it under congestion.

Keywords: software-defined networking; SDN; random forest; OpenFlow; machine learning; link flooding attacks; LFA; industrial cyber physical systems; ICPSs.

Reference to this paper should be made as follows: Deliwala, P., Jhaveri, R.H. and Ramani, S. (2022) 'Machine learning in SDN networks for secure industrial cyber physical systems: a case of detecting link flooding attack', *Int. J. Engineering Systems Modelling and Simulation*, Vol. 13, No. 1, pp.76–84.

Biographical notes: Priyanshi Deliwala completed her Bachelor's in Information and Communication Technology from the School of Engineering and Applied Science, Ahmedabad University, India. Currently working as a full stack developer in the Solvative, India. Her area of interest is in detecting pattern in SDN using analytics and machine learning algorithms.

Rutvij H. Jhaveri is currently working as a Senior Assistant Professor in the Department of Computer Science Engineering, PDPU, Gandhinagar, India. He obtained his Bachelor of Technology in Computer Engineering from the Birla Vishvakarma Mahavidyalaya, V.V. Nagar, India, MTech (R) from the SVNIT, Surat, India and completed his PhD from the CHARUSAT University, Changa, India. He pursued his post doctoral research from the Nanyang Technological University, Singapore. He has 18 years of experience in teaching and research. He authored more than 70 international publications. His research interests include network security, SDN with machine learning and data analytics.

Sagar Ramani is working as a Lecturer in the Computer Engineering Department, AVPTI Polytechninc, Rajkot. Currently, he is pursuing his PhD in the Gujarat Technological University, Ahmedabad. His area of interest is providing resiliency in SDN.

1 Introduction

The upcoming smart city model is an urban environment with new inspiration for inventive passage, monitoring healthcare, and social activities. Allowing the technology for such an environment requires a look-see of smart cities as industrial cyber physical systems (ICPSs) that incorporate new software plans of action and precise requirements for privacy, reliability, safety, mobility, and computing enormous information (Iwendi et al., 2020a; Tang et al., 2019). In order to come up with such a smart city setting requires optimised protection (Iwendi et al., 2020b; Imtiaz et al., 2021); also, there is a pressing need to develop an effective mechanism for detecting malicious applications before they exploit data (Javed et al., 2020). Furthermore, recent industrial IoT is emerging as a concept of Industry 4.0 and opens an insight into a new era in which software-defined networking (SDN) and machine learning (ML) play a vital role (Parimala et al., 2021). Presently, internet protocol (IP) networks are becoming more challenging to handle. This pattern will be more incriminating with evolving paradigms of big data applications or data centre services. A balanced, effective, and open network architecture, namely SDN, intent on reversing this situation (Hu et al., 2014). SDN is a more flexible network infrastructure with programmable devices, where new code and agreement can be introduced through software without any hardware modifications (Hu et al., 2014). In addition to this, as we are classifying malicious flow in SDN, it is essential to choose an appropriate model to avoid classification-related problems (Patel et al., 2020; ur Rehman et al., 2021; Iwendi et al., 2020c).

The data plane is receptive to providing traffic forwarding performance in the SDN context. In contrast, a centralised controller retains an overall look-see of the whole network, which can be easily configured for forwarding the traffic and optimise its performance (Kreutz et al., 2015; Shu et al., 2016). The interaction between the data plane and the controller is applied using OpenFlow (OF) (Rasool et al., 2019) policies. In the data plane, each OF-enabled system has flow tables controlled by the controller consisting of flow entries that measure the traffic route to its imminent direction (McKeown et al., 2008). All the arriving packets are compared with the flow table entries, and if the entry does not exist in the established flow, the packet which is arrived is then given to the controller for additional action. The controller's flow table is updated after studying the packet, which is arrived, and the traffic is sent to the destined path (Scott-Hayward et al., 2016). The perpetual communication between the controller and the data plane has the proficiency to cause severe network impacts and can rattle communication by causing denial-of-service (DoS). A more significant threat, a link flooding attack (LFA) occurs when the control plane is attacked with an abnormal traffic by generating congestion on the communication link (Wei and Fung, 2015; Ma et al., 2019).

LFA is a destructive type of furtive DoS attack that congests vital network connections and can separate the victim's network (ur Rasool et al., 2020). In LFA, the attacker analyses the network goal area connections to define the link. Bots are used to send the network traffic ahead of target servers on the chosen link (Kang et al., 2016). The opponent will theoretically isolate the target server from the network using this configuration without explicitly attacking the target. Recently, several surveys have been conducted that aim to reduce LFA in SDN. Some of them concentrate on revealing a counterfeit network topology that maps to the opponent (Lee et al., 2013; Kim and Shin, 2017). In contrast, others focus on link examination, which carries out analogous functioning based on vital link control. However, in their coverage of SDN vulnerabilities to LFA and their mitigation, all these surveys are restricted. The main reason why the traditional methods are not successful (Xue et al., 2014; Gkounis et al., 2016) in the sense of SDN is that SDN uses a centralised control approach to manage network congestion. The difficulty is regarding the control channel not being within reach of the expected traffic. Mitigating the effect of LFA on the control channel is a challenging operation for the same purpose.

ML allows computers to use statistical techniques to learn about a data's peculiarity or an issue and automate the result for a random dataset. ML determines to recognise and exploit the hidden patterns. ML algorithms are successfully used for fraud detection, pattern recognition, and intrusion detection systems in recent years. The existing approach utilises a deep learning technique, namely MLP, to choose suitable traffic characteristics to accurately identify a large sum of traffic data that surreptitiously acquires the connection (Rasool et al., 2019). However, when inspected using the input dataset, the research reveals numerous anomalies that could impact the system's traffic congestion efficiency. Each flow's action misbehaved, and the selected model for classifying the virulent traffic provided less precision and accuracy. Therefore, this instigated us to contribute to the implication of ML for the detection of LFA using peculiar model selection and distinct datasets. Our findings demonstrate that minimising the LFA control channel's challenge during congestion successfully uses the random forest algorithm than the deep learning MLP technique. In the paper, we map the LFA problem as an ML problem and establish a classifier that helps us attain high accuracy and precision during congestion. We select Mininet for our test-bed assessment as it is extensively accepted to deploy large networks as a practical emulator, particularly for SDN networks. Traffic congestion classification based on deep learning and random forest techniques is performed on cyber-risk analysis. Furthermore, the decision support dataset is collected from the NSL-KDD and UCI ML Library. An assessment has been conducted to appraise the multilayered perception technique classification and the random forest using the WEKA tool.

The rest of the paper is organised as follows, Section 2 presents the related work, Section 3 discusses the existing ML technique, the selection of an algorithm, and proposes a novel algorithms election for classifying the traffic during

congestion, Section 4 provides the implementation details and the results, and lastly, Section 5 concludes the paper.

2 Related work

LFA is a hazardous flooding attack capable of congesting SDN interfaces and other layers connections. In recent years, to detect different malware family variants and malicious signatures in the networks, the CNN model is used, which provides high accuracy in identifying a malware (Vasan et al., 2020). Over the last few years, it obtained a great deal of popularity as well (Bawany et al., 2017). In SDN, regarding the flow rules formed on the switches and the packets are forwarded, the controller determines the traffic route. Therefore, a traditional practice to mitigate LFA becomes baseless in those circumstances. This paper aspires to provide resistance against the control plane LFA. This sector considers the current techniques as follows:

2.1 LFA employing traffic engineering techniques

Traffic engineering is a standard procedure to mitigate an extensive array of network attacks. Hirayama et al. (2015) recommend increasing the number of traceroute packets to reveal LFA. However, distinguishing between genuine and virulent traceroute demand in the net is tough. Gillani et al. (2015) suggest a dynamic allocation of network resources using virtual network positioning during LFA. However, the approach assumes that the peer routers support end-host and that the preserver and attackers arbitrarily select the routes, regardless of the exertion taken. For mitigating LFA, network resources are continuously migrated. Liaskos et al. (2016) have suggested a mechanism for traffic engineering that uses the concepts of relational algebra for LFA mitigation. The defender redirects the traffic when the attack happens to prevent congestion. The defender continues to replicate this method until the attackers' similarity is discovered by being aware of the root cause, which is repeatedly involved in network flooding. Prime metrics for fruitful dynamic traffic engineering are available bandwidth, especially while considering congestion (Jain, 1990). However, as the number of networks grows more exhaustive and eclectic, congestion becomes more arduous to manage with more incredible velocity and substantial traffic.

2.2 LFA mitigation-based methods using SDN concepts

Few researchers exploit the advantage of providing a centrally controlled ability to monitor the switches that it authorises and their coincident flows to prevent LFA in SDN. Xiao et al. (2016) have suggested a flow nomograph inspection technique. To distinguish malicious flows, flow table inspection is performed where a detector factor uses the bloom filtering approach to observe malicious adversaries. If the link data is uncertain, it is challenging for an opponent to find the target link that precedes the occurrence of an attack. Latest suggestions on cache-like OF switch order (Ali-Ahmad et al., 2013) have spilled some sight on surmounting the experimental drawbacks of keen switching contrive for flow table sizes. In expansion, in SDN hardware implementations, assets pose a functional obstacle. In SDN architectures, several threat vectors have already been identified (Kreutz et al., 2013), as well as many security problems and vulnerabilities in OF-based networks (Klöti et al., 2013; Benton et al., 2013; Shin and Gu, 2013). However, they presume that a network operator has protected the controller and related communication channel with appropriate precautions. Bandwidth compliance is quite concerning; this is attributable to OF protocol limitations.

2.3 LFA using ML techniques

An effective DoS mitigation technique was recently suggested by Shang et al. (2017). Via table miss evaluation, surveillance of the flow rule, and packet cleaning methods prevent DoS attacks. Nevertheless, in LFA, which uses a low degree of regular traffic, such an appliance becomes invalid. For years, reinforcement learning has been practiced in networking (Boyan and Littman, 1999). Intelligent learning modules have been adapted within routers to integrate complex parameters such as congestion into the packet flow decision. Still, they could achieve minimal success because the running experiment is expensive, and the algorithm undertakes a greedy approach as network congestion increases. The sole aim of ML algorithms is to build a more stable, reliable, and effective model (Sagar et al., 2020). Therefore, ML has been used in SDN and for DDoS attack detection. The technique used by Rasool et al. (2019) is planted in ML, by which the controller uses the ML approach to determine whether or not the flows are virulent and takes further action. It has proven to yield productive results; however, because of certain limitations, this exploration's out-turns have to be seen during congestion and yields less accuracy for classifying the malicious traffic using ANN-MLP, a deep learning algorithm technique. We have found this vulnerability and suggested an explanation that can adequately mitigate LFA on the SDN control channel, even during congestion. Therefore, we coeval a novel LFA mitigation technique based on ML that uses a connection surveillance mechanism to detect LFA-induced virulent traffic flows during congestion using a ML algorithm, namely random forest. Also, resilience management in SDN congestion detection is one of the most promising factors which affect the results of the network throughput (Jhaveri et al., 2019a, 2019b). Therefore, we modified the dataset and used a distinct dataset of traffic congestion for the evaluation. We performed experiments to communicate the weakness and to elevate the results of accuracy. A detailed overview of the method or algorithm used to solve this problem is provided in the next section.

3 Proposed methodology

LFA problem and mitigating it using ML algorithms (Rasool et al., 2019) have been stated in the existing approach. In this section, we first bestow a comparison between the ANN-MLP technique and random forest so that the disparity between both is apparent. After that, we present the methodologies outlined in the existing approach for classifying the virulent traffic using a ML algorithm. We point out the model selection in these proposed methodologies. The choice of algorithm is very crucial for the type of administration it is betrothed. Some algorithms are appropriate for classifying virulent traffic but can be unsuccessful during congestion. We will present the given methodologies used in the current literature, which look perfect, but the subtle problems inherent in them are exposed during congestion analysis. Educated from the pitfalls of the methodologies discussed allow us to suggest a new method for mollifying the LFA during congestion. Table 1 summarises the software and tools used for execution and experimental assessment.

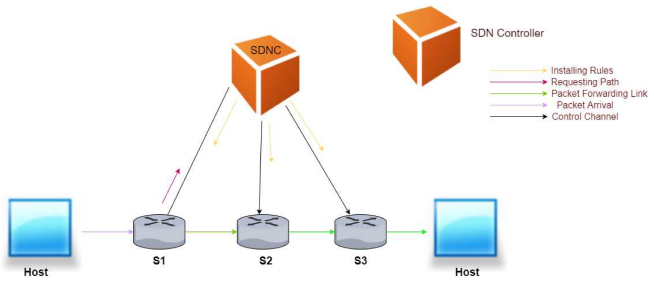
Table 1 Tools used for experimental assessment

Software	*Function*
Ubuntu	Operating system
Mininet	Network emulator
OpenFlow	Southbound interface SDN protocol
RYU	Controller for SDN
iperf	Generating TCP traffic and measuring delay

Initially, the SDN flow rule has been installed. The flow table rules handle the arriving packet, and an OF switch includes flow entries consisting of fields of priority. In Figure 1 the regular SDN trade can be seen; it can be depicted that as soon as the packet gets through on a switch, by reviewing the packet header, a query is created by the controller for the given path. The correlated can is subsequently forwarded further to the switch, and the constitution packet is transmitted to the destined route. As the attackers utilise traffic low in rate to attack the link, it clogs the traffic's illicit flow. In order to classify the licit flow with virulent flow, the ML approach has been accounted to mollify it. An ANN classification technique was selected to classify the attack accurately. They are using the input dataset to train the ANN model into classifying the arriving traffic into two classes, licit and overflow, using the WEKA tool and then performing the classification process.

Using ten-fold cross-validation, the experiment was carried out. Eight secret neurons, the combination of the input and output layers, were used in the experiment. The algorithm acquitted genuinely in terms of accuracy and time constraints on 0.3 learning rate. The experiment was regulated using a concealed layer and 0.2 momentum. It can be seen that the existing approach classifies the network influx with better accuracy. However, a similar algorithm exhibits less accuracy during congestion and while classifying the same using different datasets.

Figure 1 Normal operation of SDN (see onlne version for colours)



We saw the shortcomings of the existing method used in classifying the virulent traffic during congestion. We will partake in our comparative analysis of the ML tools and the algorithm selection how they recite our issue; we address how we gave a significant mind's eye to this decision. In this paper, we choose the random forest as preeminent to classify overflow or illicit attacks during congestion because it can be easily mollified if the virulent traffic is accurately identified. Table 2 constitutes the comparison between the ML algorithm selection of the existing approach and our approach based on their training time, classification, and learning rate. Random forest not only gives better accuracy and precision but also consumes less training time. Therefore, we selected a random forest for flood attack influx classification.

Table 2 Comparison of ML algorithms during congestion

Algorithm	*Building time*	*Accuracy*	*Precision*
Random forest	249 sec.	98%	99%
ANN-MLP	352 sec.	93%	92%

Using the UCI ML Library and the NSL-KDD dataset, the random forest classification module constructs a training model. It was composed of 22 features (Dua and Taniskidou, 2019) 14 features have been decided for building a traffic model after careful investigation of the dataset and the problem put forward. The significance behind choosing the 14 features was to remove the features which played an insignificant role in classifying the network flows. By reducing the input features, we get a better understanding of the dataset. The feature was selected based on its importance score for the given problem statement. More the score; more the significance. It is by using the Shapley values to estimate how each function contributes to the prediction. It can be installed and used easily (pip install shap) with scikit-learn random forest. Out of the 22 features, the 14 features that we selected contribute significantly. For example, if the link is being flooded or utilises more or less bandwidth than the link capacity, the bandwidth lost and the total bandwidth attribute will educate us. Thus, such an attribute selection enhances the overall output accuracy. Thus, helping in removing the outliers. Information on the characteristics of the dataset is provided inside Table 3. At the posterior, it manages the procedure of classification based on the trained model. The output that consists of the licit flows is exhibited.

The steps involved in the module for detecting the virulent traffic are provided in the algorithm. The network offers the best routes that fit each state by the controller during the module's preparation.

Table 3 Features used in the analysis and in the obtained statistics

Sr. no.	*Attribute*
1	Node
2	Utilised bandwidth
3	Packet drop rate
4	Full bandwidth
5	Packet lost rate
6	Lost byte rate
7	Rate of packet received
8	Bandwidth used
9	Bandwidth lost
10	Size of packet
11	Packet received
12	Packet lost
13	Byte transmitted
14	Flow status

Algorithm 1 Algorithm using random forest classifying LFA

Necessitate: Network flood NF, dataset d
Secure: Flood class: Illicit, Licit
1 Get network flood
2 **for** Flood nf ∈ NF (network flood), **do**
- Get flood estimation
- Pry attributes

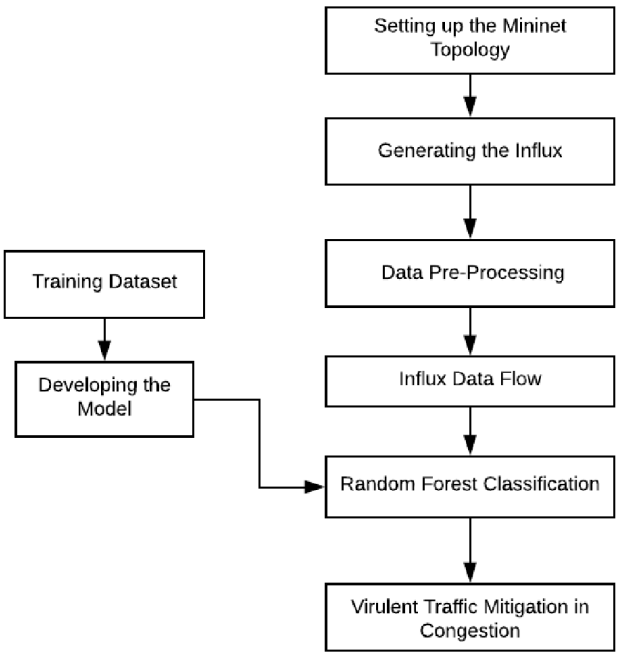
3 **end for**
4 Pre-processing and cleaning d
5 Train random forest model using d
6 Distinguish flood employing random forest
7 Remove classification
8 **return** Flood class: Illicit, Licit

As ML model identifies malicious and benign flow, all the 14 features play a vital role in making a suitable decision. All packets related features such as packet drop rate, rate of the packet received, the packet received, packet lost, and packets' size show that if congestion increases in the network, packet drop rate also increases. This happens due to increased attackers injecting the network's malicious flow, increasing the packet drop rate. Therefore, these feature sets elevate the ML model's accuracy to identify the type of flow. While bandwidth-related feature sets such as bandwidth lost, bandwidth used, total bandwidth, flow status show that if the number of attackers increases, bandwidth consumption also increases. Therefore, these features are also helpful to identify the malicious and benign flow in the network. Once the machine learns the network topology, it can optimise the routing flows of the traffic. Since the congestion and traffic configuration account in the network, the traffic is routed in the network. The network state is assimilated at the level of the SDN controller. Figure 2 shows the flow of the experimental setup. We start by establishing a Mininet network and procreating a set traffic flow. Moreover, switch, and port estimates are attained using the RYU application. A standalone computer was used to execute the virtual network topology in an emulated network environment. Ubuntu was put-to-use as our host machine, and the RYU controller, coded in the Python language. We have presented the parameters in Table 4 of the traffic procreation during normal and congestion conditions. To determine the links in common to related OF devices, the controller employs link layer discover protocol messages. In the controller data cache, the topology manager cache and the information are supervised for the nodes and links of the traffic initiated.

Table 4 Parameters for traffic procreation

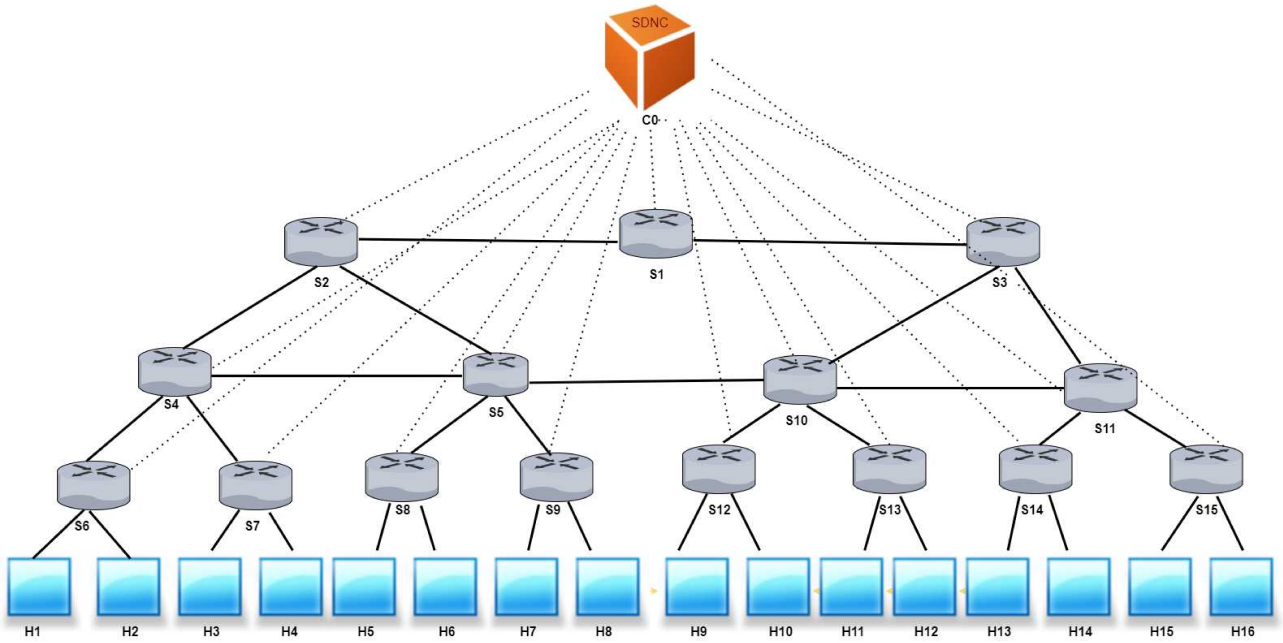
Parameters	*Value/function*
Number of hosts	16
Number of switches	15
Port number of controller	6653
Link capacity	1,000 Mbps
Delay estimation time	10 s
Type of traffic	TCP
Traffic generation	iperf

Figure 2 Flowchart of the implementation steps



The protocol of classification is explained in the algorithm. The algorithm captures network flood set NF input and the dataset used for training (d). After the extraction of all the features, the traffic congestion estimation captures the traffic flood and then performs pre-processing. Later, the classifier is trained using the random forest algorithm; at the end, the same is used to classify the captured network estimation into the two classes, namely, illicit and licit. The input dataset was downloaded for the flooding attack (Xiao et al., 2016). And the NSL-KDD dataset was downloaded for the traffic congestion.

Figure 3 Procreated topology using Mininet and OF controller (see online version for colours)



We trained and tested the traffic dataset after the circumspection of the dataset and the aspired problem. The way that each flow behaves is examined, and when the flow is misbehaving is noted. The random forest classifier model carries out its efficacy by classifying the network flow into two classes: illicit and licit. The topology of the system was devised, as shown in Figure 3. Three levels of tree topology with the corresponding 16 hosts and 12 OF switches are depicted in the figure. Ubuntu operating system is used where the topology is created in Mininet as shown in Figure 3, the experimental setup of network exits of 12 OF switches and 16 hosts, where all the switches and host are connected to the controller with an association link capacity of 1,000 Mbps. After that, traffic is initiated towards the marked hosts. The traffic is initiated at a low rate first, and then the congestion situation is created. In order to construct and estimate network traffic flood, an open-source application, iperf is used. Using the iperf – s command, the TCP server connection is initiated, and the server will listen to TCP at port 5001. The instruction iperf – s – c 10.0.0.1 – t5, some host from the given system will then connect to the main server, which records the linked server's IP address in the given time of the demanded link. At the end of the link, the bandwidth is reflected on both client and server terminals. The server continues listening to the port after a TCP connection has been completed, providing an opening for some other host to connect to the server. The link bandwidth is examined from the experiments performed by initiating the traffic for congestion. Traffic is controlled for each node sent or attacked in the Mininet by commencing a separate terminal in Ubuntu machine.

It can be noted that with the increase in attack traffic, there is an increase in packet drop rate. A drastic change in the packet drop rate was acknowledged when the attackers' figure rose to 12, indicating an exponential rise between the packet drop rate and the various attackers. An empirical extraction of flows was captured for each node in the network. Data pre-processing was implemented after the traffic was drawn out, and statistics were extracted for the respective flows. The model was constructed using the training set, and the trained model was further used to classify statistics into a benevolent and virulent flow.

4 Results

We address and evaluate the outcomes of the experiments in this section. Using three different precision assessment criteria, we conduct detailed experiments and test them. Accuracy, recall, and the F1 score ranking is the three metrics that assess ML algorithms' performance. The assessment is further carried out using appropriate data partitioning techniques, attribute selection, and various classification classifiers.

Three accuracy assessment criteria, mainly precision, recall, and F1 ranking, were put to use. The calculation of how near the expected estimate and the actual estimate is defined as precision. When fractionated by the integral number of recorded flows, the value is the number of correlated flows retrieved. In Table 4, the formula for all three metrics is given. The calculation of accuracy evaluation metrics during congestion is shown in Figure 4. The two classes' accuracy is more than 95% during congestion while using random forest than in ANN-MLP. The recall is elucidated as the integral part of classified relevant flows fractionate by the essential role of flows retrieved. The recall measure's overall accuracy was about 95%. The F1 score is also a significant indicator of assessment since it blends both accuracy and recall values.

And the F1 score is 94%, which is comparatively quite more than ANN-MLP, as shown in Figure 5.

Table 5 Comparison of ML algorithms during congestion

Metric	*Formula*
Precision	$\frac{True\ positive}{True\ positive+False\ positive}$
Recall	$\frac{True\ positive}{True\ positive+False\ negative}$
F1 score	$2 \times \frac{Precision \times Recall}{Precision+Recall}$

The observations and inspection indicate that we were able to identify the traffic flow during congestion more accurately. Contingent upon the values of the classification, flood reduction was proficient in eliminating the flood traffic. Overall, if the intruder is efficient and can redirect high flood traffic volumes, it can be inferred that the system's usable bandwidth would be dropped, which can be remarked from Figure 6. Similarly, the packet drop rate will also appear to increment if the flooding attack is increased, as shown in Figure 7, severely influencing the network's licit traffic. It should also be remembered that we constructively classify a flow convoluted in the network's flooding.

Figure 4 Evaluation of the random forest classifier during congestion using accuracy metrics (see online version for colours)

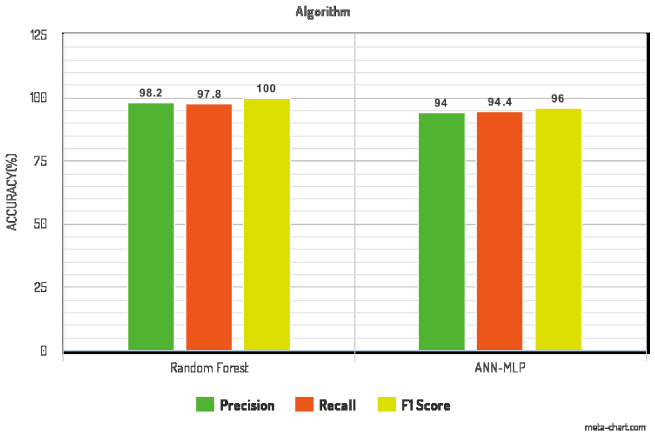
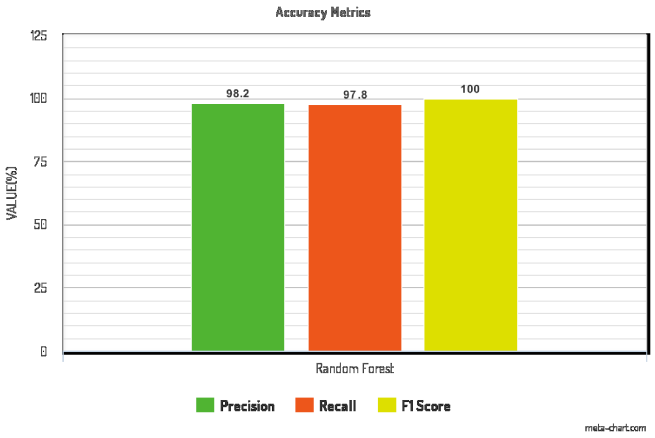


Figure 5 Classifier evaluation using algorithm selection during congestion (see online version for colours)



We have taken 1,500 tuples dataset, and we split the dataset into two partitions concerning training and testing, i.e., 60% training and 40% testing. As the training rate of data was escalated, beginning with 50% to more than 20%, there was a substantial improvement in the assessment metrics' accuracy. The accuracy metrics quickly improved to over 95%, which was around 87%, while 50% of training data was used. In order to get better results, the juxtaposition offers perception into choosing an unbiased split of training and testing data.

Figure 6 Consequences of LFA on the link bandwidth congestion (see online version for colours)

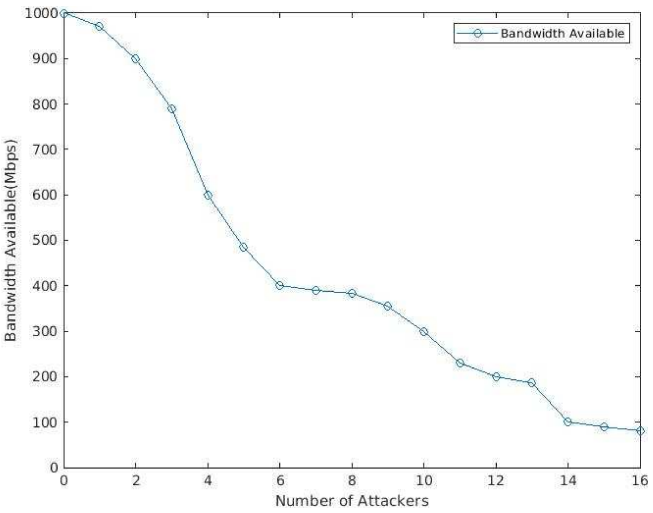
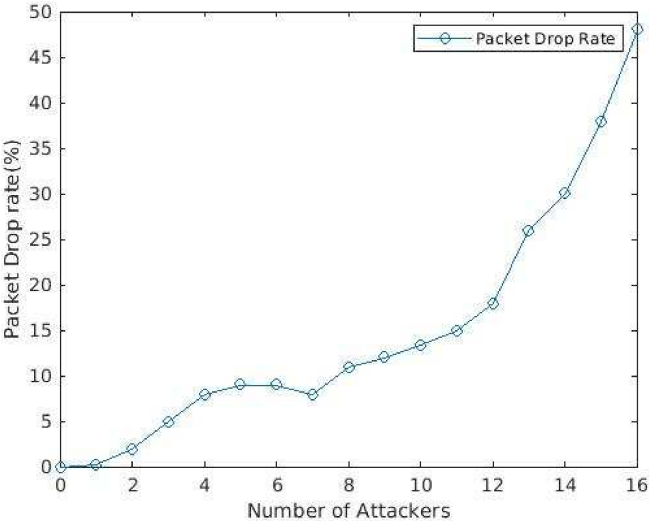


Figure 7 Consequences of LFA on the packet drop rate (see online version for colours)



As depicted in Figure 6, as the number of attackers increases, the bandwidth availability decreases. This happens because as the number of attackers increases, they insert malicious flow in the network. As a result, available bandwidth is consumed, and the availability of free bandwidth decreases. While in Figure 7, it is observed that as the number of attackers increases, the packet drop rate eventually increases. As the number of attackers

increases, malicious flows also increase due to injection in the attackers' network. Therefore, congestion is increased in the network, which causes a higher packet drop rate. If we observe the ML algorithms, due to their strong predictive efficiency and outstanding understanding, random forest is adopted to classify synchronous time data.

Due to their solid predictive efficiency and exceptional understanding, random forest is used to classifying synchronous time data. In that case, consequently, it was well suited if attributes are not depended on linearly. The features selected for the proposed model, essentially maximum bandwidth and the packet drop rate, were not linearly dependent on each other. There are several layers in the MLP deep learning methodology, together with the input sensory layer, the output layer, and the hidden layers that function together to draw out the problematic primary features. However, the model's training period is very high, and we found that the accuracy and value of recall fell significantly during congestion. The classification's comprehensive accuracy was of primary significance since it involved classifying the two classes of network traffic. Therefore, the random forest algorithm was ideal for our requirements, as it worked expertly and produced reasonably upstanding collective results.

5 Conclusions

Well-designed CPS will improve innovative city comfort, revitalise urban manufacturing sites, and increase optimised security chances. For complex data centres, businesses, and, more recently, WAN environments, SDN continues to grow in demand. The arrangement of the separate data plane, control plane, and application plane, also the control channel, allows flexibility for complex networks in administration and supplication creation. We also supported the necessity to tackle SDN-based LFA in this paper with the aid of an assessment that illustrates the effect of a victorious attack. We analyse the control channel statistics, and the network traffic is classified wielding a random forest strategy during congestion and compare it with multi-layer perception for deep learning. Finally, the potential to efficiently decrease the flows which are classified and promote the network's smooth functioning is demonstrated. Based on the assessment, it can be concluded that we can correctly recognise traffic flows during congestion that exhibit LFA features and effectively mollifies the attack more accurately using the random forest algorithm. A drawback, however, is the additional difficulty of using an ML strategy. An expansion of the proposed structure is underway, and future studies will present the comprehensive design and evaluation.

References

Ali-Ahmad, H. et al. (2013) 'CROWD: an SDN approach for DenseNets', in *2013 Second European Workshop on Software Defined Networks*, pp.25–31.

Bawany, N.Z., Shamsi, J.A. and Salah, K. (2017) 'DDoS attack detection and mitigation using SDN: methods, practices, and solutions', *Arab. J. Sci. Eng.*, Vol. 42, No. 2, pp.425–441.

Benton, K., Camp, L.J. and Small, C. (2013) 'OpenFlow vulnerability assessment', in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, pp.151–152.

Boyan, J. and Littman, M. (1999) 'Packet routing in dynamically changing networks: a reinforcement learning approach', *Adv. Neural Inf. Process. Syst.*, Vol. 6.

Dua, D. and Taniskidou, E.K. (2019) *UCI Machine Learning Repository*, University of California, School of Information and Computer Science, Irvine, CA [online] http://archive.ics.uci.edu/ml.

Gillani, F., Al-Shaer, E., Lo, S., Duan, Q., Ammar, M. and Zegura, E. (2015) 'Agile virtualized infrastructure to proactively defend against cyber attacks', in *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp.729–737.

Gkounis, D., Kotronis, V., Liaskos, C. and Dimitropoulos, X. (2016) 'On the interplay of link-flooding attacks and traffic engineering', *SIGCOMM Comput. Commun. Rev.*, May, Vol. 46, No. 2, pp.5–11.

Hirayama, T., Toyoda, K. and Sasase, I. (2015) *Fast Target Link Flooding Attack Detection Scheme by Analyzing Traceroute Packets Flow*, DOI: 10.1109/WIFS.2015.7368594.

Hu, F., Hao, Q. and Bao, K. (2014) 'A survey on software-defined network and OpenFlow: from concept to implementation', *IEEE Commun. Surv. Tutorials*, Vol. 16, No. 4, pp.2181–2206.

Imtiaz, S.I., ur Rehman, S., Javed, A.R., Jalil, Z., Liu, X. and Alnumay, W.S. (2021) 'DeepAMD: detection and identification of Android malware using high-efficient deep artificial neural network', *Future Generation Computer Systems*, Vol. 115, pp.844–856, ISSN: 0167-739X [online] https://doi.org/10.1016/j.future.2020.10.008.

Iwendi, C. et al. (2020a) 'Key split watermark: zero watermarking algorithm for software protection against cyber-attacks', *IEEE Access*, Vol. 8, pp.72650–72660, DOI: 10.1109/ACCESS.2020.2988160.

Iwendi, C., Maddikunta, P.K.R., Gadekallu, T.R., Lakshmanna, K., Bashir, A. and Piran, M.J. (2020b) 'A metaheuristic optimization approach for energy efficiency in the IoT networks', *Softw. Pract. Exp.*, pp.1–14, DOI: 10.1002/spe.2797.

Iwendi, C., Khan, S., Anajemba, J.H., Mittal, M., Alenezi, M. and Alazab, M. (2020c) 'The use of ensemble models for multiple class and binary class classification for improving intrusion detection systems', *Sensors (Basel)*, April, Vol. 20, No. 9, p.2559.

Jain, R. (1990) 'Congestion control in computer networks: issues and trends', *IEEE Netw.*, May, Vol. 4, No. 3, pp.24–30.

Javed, A.A.R., Jalil, Z., Moqurrab, S.A., Abbas, S. and Liu, X. (2020) 'Ensemble Adaboost classifier for accurate and fast detection of botnet attacks in connected vehicles', *Trans. Emerging Tel. Tech.*, p.e4088 [online] https://doi.org/10.1002/ett.4088.

Jhaveri, R.H., Tan, R., Easwaran, A. and Ramani, S.V. (2019a) 'Managing industrial communication delays with software-defined networking', in *2019 IEEE 25th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*.

Jhaveri, R.H., Tan, R. and Ramani, S.V. (2019b) 'Real-time QoS-aware routing scheme in SDN-based robotic cyber-physical systems', in *2019 IEEE 5th International Conference on Mechatronics System and Robots (ICMSR)*, pp.18–23.

Kang, M., Gligor, V. and Sekar, V. (2016) *SPIFFY: Inducing Cost-Detectability Tradeoffs for Persistent Link-Flooding Attacks*, DOI: 10.14722/ndss.2016.23147.

Kim, J. and Shin, S. (2017) 'Software-defined HoneyNet: towards mitigating link flooding attacks', in *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pp.99–100.

Klöti, R., Kotronis, V. and Smith, P. (2013) 'OpenFlow: a security analysis', in *2013 21st IEEE International Conference on Network Protocols (ICNP)*, pp.1–6.

Kreutz, D., Ramos, F.M.V. and Verissimo, P. (2013) 'Towards secure and dependable software-defined networks', in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, pp.55–60.

Kreutz, D., Ramos, F.M.V., Veríssimo, P.E., Rothenberg, C.E., Azodolmolky, S. and and Uhlig, S. (2015) 'Software-defined networking: a comprehensive survey', *Proc. IEEE*, January, Vol. 103, No. 1, pp.14–76.

Lee, S.B., Kang, M.S. and Gligor, V.D. (2013) 'CoDef: collaborative defense against large-scale link-flooding attacks', in *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, pp.417–428.

Liaskos, C., Kotronis, V. and Dimitropoulos, X. (2016) 'A novel framework for modeling and mitigating distributed link flooding attacks', in *IEEE INFOCOM 2016 – The 35th Annual IEEE International Conference on Computer Communications*, pp.1–9.

Ma, X., Li, J., Tang, Y., An, B. and Guan, X. (2019) 'Protecting internet infrastructure against link flooding attacks: a techno-economic perspective', *Inf. Sci. (Ny).*, Vol. 479, pp.486–502.

McKeown, N. et al. (2008) 'OpenFlow: enabling innovation in campus networks', *SIGCOMM Comput. Commun. Rev.*, March, Vol. 38, No. 2, pp.69–74.

Parimala, M. et al. (2021) *Fusion of Federated Learning and Industrial Internet of Things: A Survey*.

Patel, H., Rajput, D.S., Reddy, G.T., Iwendi, C., Bashir, A.K. and Jo, O. (2020) 'A review on classification of imbalanced data for wireless sensor networks', *Int. J. Distrib. Sens. Networks*, Vol. 16, No. 4, p.1550147720916404.

Rasool, R., Ashraf, U., Ahmed, K., Wang, H., Rafiq, W. and Anwar, Z. (2019) 'CyberPulse: a machine learning based link flooding attack mitigation system for software defined networks', *IEEE Access*, p.1, DOI: 10.1109/ACCESS.2019.2904236.

Sagar, R., Jhaveri, R. and Borrego, C. (2020) 'Applications in security and evasions in machine learning: a survey', *Electronics*, Vol. 9, No. 1, p.97.

Scott-Hayward, S., Natarajan, S. and Sezer, S. (2016) 'A survey of security in software defined networks', *IEEE Commun. Surv. Tutorials*, Vol. 18, No. 1, pp.623–654.

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

Shin, S. and Gu, G. (2013) 'Attacking software-defined networks: a first feasibility study', in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, pp.165–166.

Shu, Z. et al. (2016) 'Traffic engineering in software-defined networking: measurement and management', *IEEE Access*, Vol. 4, pp.3246–3256, DOI: 10.1109/ACCESS.2016.2582748.

Tang, M., Alazab, M. and Luo, Y. (2019) 'Big data for cybersecurity: vulnerability disclosure trends and dependencies', *IEEE Trans. Big Data*, Vol. 5, No. 3, pp.317–329.

ur Rasool, R., Wang, H., Ashraf, U., Ahmed, K., Anwar, Z. and Rafique, W. (2020) 'A survey of link flooding attacks in software defined network ecosystems', *J. Netw. Comput. Appl.*, Vol. 172, p.102803, DOI: 10.1016/j.jnca.2020.102803.

ur Rehman, S., Khaliq, M., Imtiaz, S.I., Rasool, A., Shafiq, M., Javed, A.R., Jalil, Z. and Bashir, A.K. (2021) 'DIDDOS: an approach for detection and identification of distributed denial of service (DDoS) cyberattacks using gated recurrent units (GRU)', *Future Generation Computer Systems*, Vol. 118, pp.453–466, ISSN: 0167-739X [online] https://doi.org/10.1016/j.future.2021.01.022.

Vasan, D., Alazab, M., Wassan, S., Naeem, H., Safaei, B. and Zheng, Q. (2020) 'IMCFN: image-based malware classification using fine-tuned convolutional neural network architecture', *Comput. Networks*, Vol. 171, p.107138, DOI: 10.1016/j.comnet.2020.107138.

Wei, L. and Fung, C. (2015) 'FlowRanger: a request prioritizing algorithm for controller DoS attacks in software defined networks', in *2015 IEEE International Conference on Communications (ICC)*, pp.5254–5259.

Xiao, P., Li, Z., Qi, H., Qu, W. and Yu, H. (2016) 'An efficient DDoS detection with bloom filter in SDN', in *2016 IEEE Trustcom/BigDataSE/ISPA*, pp.1–6.

Xue, L., Luo, X., Chan, E.W.W. and Zhan, X. (2014) 'Towards detecting target link flooding attack', in *28th Large Installation System Administration Conference (LISA14)*, pp.90–105.