International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)
ISSN: 0976-1353 Volume 28 Issue 7 – JULY 2021.

DDoS ATTACK DETECTION IN SOFTWARE DEFINED NETWORKS USING GENETIC ALGORITHM AND MACHINE LEARNING

[1] Dr.R.Murugeswari, [2] T.Naresh Kumar Reddy, [3] T.Prasanth Rayalu
[1]Associate Professor, Department of Computer Science and Engineering, Kalasalingam Academy of Research and Education, Anand Nagar, Krishnankoil, India, r.murugeswari@klu.ac.in
[2] Department of Computer Science and Engineering, Kalasalingam Academy of Research and Education, Anand Nagar, Krishnankoil, India,thoorpintinareshkumarreddy@gmail.com
[3] Department of Computer Science and Engineering, Kalasalingam Academy of Research and Education, Anand Nagar, Krishnankoil, India, prasanthrayalu@gmail.com

.

Abstract : Software-Defined Network (SDN) has been an emerging industry, which provides network operators control on network infrastructure. SDN controller is responsible for running various network applications and maintaining several network services and functionalities. SDN faces many security threats, the most important one is Distributed Denial of Services (DDoS). This proposed system detects the network attack on SDN using novel techniques. The work carried in two phases. First, Genetic algorithm is applied to choose the important features, the second applying machine learning algorithm like regression techniques to detect the type of attack. The proposed system uses KDD dataset for it evaluation of the above techniques. The proposed work is implemented in Python 3.6.4 with libraries scikit-learn, pandas, matplotlib and other mandatory libraries. The proposed work is compared to SVM and Logistic regression shows better accuracy compared to SVM in detection.

Keywords: *Distributed Denial of Services, Software-Defined Network.*Introduction

Introduction:

Recently the Software Defined Networks (SDN) paradigm has gained significant interest from many researchers. The SDN paradigm offers a greater potential to provide a secure, flexible, and reliable network system. Separation of the control plane from the underlying infrastructure layer is the main innovation behind SDN. The centralized. controller manages the packet-forwarding devices that need to be configured via a well-designed interface like Open- Flow. In SDN, the network devices like switches have only forwarding logic, whereas the control logic and decision-making ability are softwarized at the controller. This allows the controller to instruct the switches with new network policies, and underlying devices start to follow the policies maintain in the flow table. When a packet arrives at a switch, it checks its flow table, and if the flow matches, it forwards the packet to the destination. If no match founds in the flow table, OpenFlow enabled switch sends control packet to the controller for making an appropriate decision. The controller can handle multiple flow tables maintained by OpenFlow switch, consequently achieving programmability in the control layer of SDN. According to the controller policy, the flow tables can serve as a switch, firewall or router that exhibit similar roles. Despite all these impressive innovations, various architectural components pose additional security threats to SDN. As far as different issues to be addressed, the security of SDN is considered as the highest concern. Among many security threats, one of the critical security issues is Distributed Denial of Service (DDoS). The main aim of this attack is to make computing resources unavailable to the legitimate users. This attack is usually caused by more than one bot, penetrated by software from malicious code. As the initial process is simple, the DDoS attack can quickly spread and cause massive damage to the network, but the defend process is very troublesome. Hence, it is essential to impose certain security rules on the controller. Therefore, an efficient detection technique and mitigation rules must be designed for future network architecture like SDN. Since the controller is the central intelligent part of the SDN, several techniques like neural network and machine learning can be used to leverage network security.

Different classes of Attacks:

Denial of Service(DoS)

An attacker tries to prevent legitimate users from using a service. For example, SYN flood, Smurf and teardrop.

User to Root (U2R)

An attacker has local access to the victim machine and tries to gain super-user privilege. For example, buffer overflow attacks.

Remote to Local (R2L)

An attacker tries to gain access to victim machine without having an account on it. For example, password guessing attack.

Probe

An attacker tries to gain information about the target host. For example, port-scan and ping-sweep.

I. RELATED WORKS

Sharma et al. [1] proposed cutting-edge cloud frameworks requires a paradigm shift in regards to how they are built and managed. Traditional management and control platforms face significant challenges in terms of security, reliability, and flexibility that these cutting-edge frameworks must deal with. On the other hand, Distributed Denial of Service (DDoS) attacks have become a weapon of choice for cyber-terrorists, cyber-extortionists, and hackers.

D. Yin, L, et al. [2] explained the spread of Internet of Things' (IoT) applications, security has become extremely important. A recent distributed denial-of-service (DDoS) attack revealed the ubiquity of vulnerabilities in IoT, and many IoT devices unwittingly contributed to the DDoS attack. The emerging software-defined anything (SDx) paradigm provides a way to safely manage IoT devices.

Sicari et al. [3] described Denial of Service (DoS) attack represents until now a relevant problem in Internet-based contexts. In fact, it is both difficult to recognize and to counteract. Along with the adoption and diffusion of Internet of Things (IoT) applications, such an issue has become more urgent to solve, due to the presence of heterogeneous data sources and to the wireless nature of most communications data processing, provide useful services to the interested users.

Y. Meidan et al [4] presented the proliferation of IoT devices that can be more easily compromised than desktop computers has led to an increase in IoT-based botnet attacks. To mitigate this threat, there is a need for new methods that detect attacks launched from compromised IoT devices and that differentiate between hours- and milliseconds-long IoT-based attacks. In this article, we propose a novel network-based anomaly detection method for the IoT the evaluation results demonstrated our proposed methods ability to accurately and instantly detect the attacks as they were being launched from the compromised IoT devices that were part of a botnet.

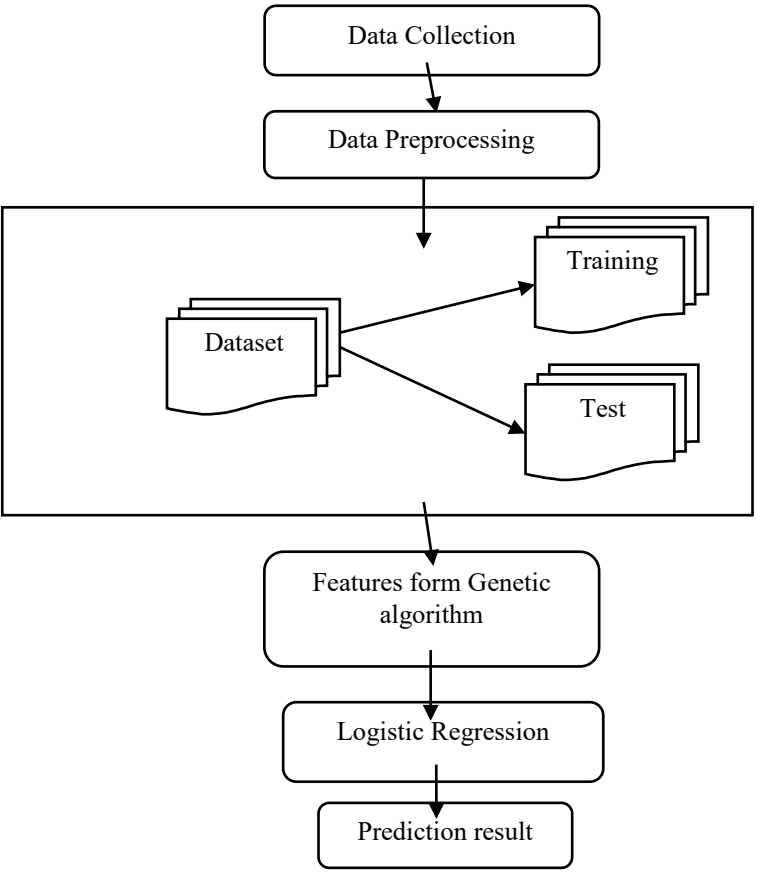
Mehmood et al [5] explained that Internet of Things (IoT) makes physical objects and devices interact with each other through wireless technologies. IoT is expected to deliver a significant role in our lives in near future. However, at the current stage, IoT is vulnerable to various kinds of security threats just like other wired and wireless networks. Our work mainly focuses on protecting an IoT infrastructure from distributed denial-of-service attacks generated by the intruders.

H. Wang, L. Xu et al[6] described that SDN-specific attack, i.e., data-to-control plane saturation attack, which overloads the infrastructure of SDN networks. In this attack, an attacker can produce a large amount of table-miss packet_in messages to consume resources in both control plane and data plane. To mitigate this security threat, we introduce an efficient, lightweight and protocol-independent defense framework for SDN networks. To preserve network policy enforcement, proactive flow rule analyzer dynamically derives proactive flow rules by reasoning the runtime logic of the SDN/OpenFlow controller and its applications

R. Braga et al[7] presented a platform to efficiently detect and rapidly respond to the DDoS attack in VNs based on software-defined networking (SDN). The

proposed platform not only contains the trigger mechanism based on the message of OpenFlow protocol (i.e., PACKET_IN message) for a response not timely but also involves a flow feature extraction strategy based on the multi-dimensional information.

SYSTEM ARCHITECTURE



Data collection:

The data collection process involves the selection of quality data for analysis. Here we used Intrusion dataset KDD dataset website dataset taken from uci.edu for machine learning implementation. The job of a data analyst is to find ways and sources of collecting relevant and comprehensive data, interpreting it, and analyzing results with the help of statistical techniques.

Data preprocessing

The purpose of preprocessing is to convert raw data into a form that fits machine learning. Structured and clean data allows a data scientist to get more precise results from an applied machine learning model. The technique includes data formatting, cleaning, and sampling.

Dataset splitting

A dataset used for machine learning should be partitioned into three subsets — training, test, and validation sets.

Training set. A data scientist uses a training set to train a model and define its optimal parameters it has to learn from data.

Test set. A test set is needed for an evaluation of the trained model and its capability for generalization. The latter means a model's ability to identify patterns in new unseen data after having been trained over a training data. It's crucial to use different subsets for training and testing to avoid model overfitting, which is the incapacity for generalization we mentioned above.

Model training

After a data scientist has preprocessed the collected data and split it into train and test can proceed with a model training. This process entails "feeding" the algorithm with training data. An algorithm will process data and output a model that is able to find a target value (attribute) in new data an answer you want to get with predictive analysis. The purpose of model training is to develop a model.

Model evaluation and testing

The goal of this step is to develop the simplest model able to formulate a target value fast and well enough. A data scientist can achieve this goal through model tuning. That's the optimization of model parameters to achieve an algorithm's best performance.

EXPERIMENTAL RESULTS

1) DATASET DESCRIPTION

The datasets contain a total of 24 training attack types, with an additional 14 types in the test data only.

The dataset variable names are described below

feature name	***description***	***type***
duration	length (number of seconds) of the connection	continuous
protocol_type	type of the protocol, e.g. tcp, udp, etc.	discrete

service	network service on the destination, e.g., http, telnet, etc.	discrete
src_bytes	number of data bytes from source to destination	continuous
dst_bytes	number of data bytes from destination to source	continuous
flag	normal or error status of the connection	discrete
land	1 if connection is from/to the same host/port; 0 otherwise	discrete
wrong_fragment	number of ``wrong'' fragments	continuous
urgent	number of urgent packets	continuous

Table 1: Basic features of individual TCP connections.

feature name	***description***	***type***
hot	number of ``hot'' indicators	continuous
num_failed_logins	number of failed login attempts	continuous
logged_in	1 if successfully logged in; 0 otherwise	discrete
num_compromised	number of ``compromised'' conditions	continuous
root_shell	1 if root shell is obtained; 0 otherwise	discrete
su_attempted	1 if ``su root'' command attempted; 0 otherwise	discrete
num_root	number of ``root'' accesses	continuous
num_file_creations	number of file creation operations	continuous
num_shells	number of shell prompts	continuous
num_access_files	number of operations on access control files	continuous
num_outbound_cmds	number of outbound commands in an ftp session	continuous
is_hot_login	1 if the login belongs to the ``hot'' list; 0 otherwise	discrete
is_guest_login	1 if the login is a ``guest''login; 0 otherwise	discrete

Table 2: Content features within a connection suggested by domain knowledge.

feature name	*description*	*type*
count	number of connections to the same host as the	continuous
	Note: The following features refer to these same	
serror_rate	% of connections that have ``SYN'' errors	continuous
rerror_rate	% of connections that have ``REJ'' errors	continuous
same_srv_rate	% of connections to the same service	continuous
diff_srv_rate	% of connections to different services	continuous
srv_count	number of connections to the same service as the	continuous
	Note: The following features refer to these same service	
srv_serror_rate	% of connections that have ``SYN'' errors	continuous
srv_rerror_rate	% of connections that have ``REJ'' errors	continuous
srv_diff_host_rate	% of connections to different hosts	continuous

Table 3: Traffic features computed using a two-second time window.

Genetic Algorithm for Feature selection problem

Step 1: Generate random population of n chromosomes
Step 2: Evaluate the fitness f(x) of each chromosome x in the population
Step 3: Create a new population by repeating following steps until the new population is complete
Step 4: Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
Step 5: With a crossover probability cross over the parents to form new offspring.
If no crossover was performed, offspring is the exact copy of parents.
Step 6: With a mutation probability mutate new offspring at each locus
Step 7: Place new offspring in the new population
Step 8: Use new generated population for a further run of the algorithm
Step 9: If the end condition is satisfied, stop, and return the best solution in current population
Step 10: Go to step 2

The following code is used for training and prediction through Logistic regression

```
#spliting the dataset into training set
                and test set
      x_train, x_test, y_train, y_test =
train_test_split(x,y,test_size = 0.25,
            random_state =0 )


          model = LogisticRegression()
          model.fit(x_train, y_train)


        #predicting the tests set result
          y_pred = model.predict(x_test)
```

The proposed work is implemented in Python 3.6.4 with libraries scikit-learn, pandas, matplotlib and other mandatory libraries. The training dataset of KDD contains 125973 rows. The test dataset contains 22543. Genetic algorithm is applied for feature selection and Machine learning algorithm is applied such as Support Vector machine and Logistic regression. We used these machine learning algorithm and indentified intrusion. The result shows that intrusion detection is efficient using Logistic regression algorithm.

The following table shows the accuracy arrived in our experimental study.

Algorithm	Accuracy (%)
Logistic regression	65.73
Support Vector machine	61.08

Table: Experimental Results of proposed system

The below figure shows the accuracy comparison of our proposed work, in which Logistic regression method outperforms.

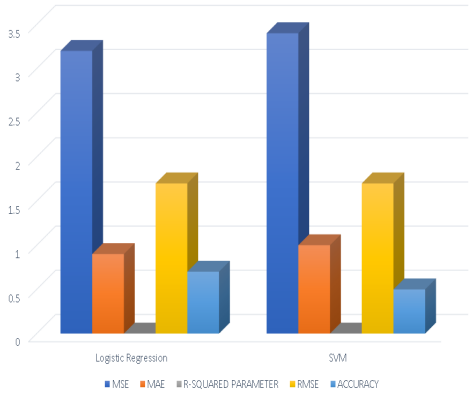
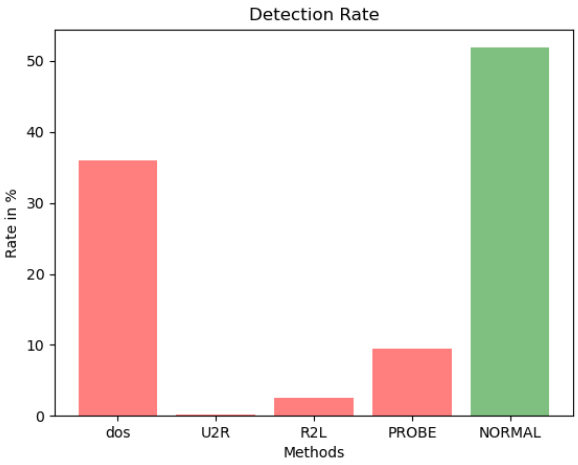


Figure: Evaluation metric of Logistic regression and svm algorithm

This is graphical notation of the data given by the system. This phase of implementation will shows the effectiveness of the proposed system through pictorially in the order to better understand of proposed system.



Detection rate of intrusion in whole data set.

CONCLUSION

The proposed work is a novel DDoS detection and mitigation framework for an SDN system. For detection purposes, the multi-layer Support Vector Machine (SVM) has used as the classifier. For better accuracy and to lessen the testing time, Genetic algorithm (GA) has been employed in this model. This technique is used to extract the principal features from the DDoS dataset; GA is used for selecting suitable parameters for SVM classifier. Furthermore, the experimental outcome exhibits that on DDoS dataset, the

proposed model performs effectively. The accuracy of the proposed model is 66%, which is better than the rest of the model.

REFERENCES

[1] Y. Zhang, L. Cui, W. Wang, and Y. Zhang, ``A survey on software defined networking with multiple controllers," J. Netw. Comput. Appl., vol. 103, pp. 101118, Feb. 2018.

[2] P. Visu, L. Lakshmanan, V. Murugananthan, and M. V. Cruz, ``Software defined forensic framework for malware disaster management in Internet of Thing devices for extreme surveillance," Comput. Commun., vol. 147, pp. 1420, Nov. 2019.

[3] E. Molina and E. Jacob, ``Software-defined networking in cyber-physical systems: A survey," Comput. Electr. Eng., vol. 66, pp. 407419, Feb. 2018.

[4] A. Mondal, S. Misra, and I. Maity, ``AMOPE: Performance analysis of OpenFlow systems in software-dened networks," IEEE Syst. J., vol. 14, no. 1, pp. 124131, Mar. 2020.

[5] M. Conti, C. Lal, R. Mohammadi, and U. Rawat, ``Lightweight solutions to counter DDoS attacks in software dened networking," Wireless Netw., vol. 25, no. 5, pp. 27512768, Jul. 2019.

[6] C. B. Zerbini, L. F. Carvalho, T. Abrao, and M. L. Proença, ``Wavelet against random forest for anomaly mitigation in software-dened networking," Appl. Soft Comput., vol. 80, pp. 138153, Jul. 2019.

[7] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, ``Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments," Comput. Netw., vol. 62, pp. 122136, Apr. 2014.

[8] R. T. Kokila, S. T. Selvi, and K. Govindarajan, ``DDoS detection and analysis in SDN-based environment using support vector machine classifier," in Proc. 6th Int. Conf. Adv. Comput. (ICoAC), Dec. 2014, pp. 205210.

[9] R. Braga, E. Mota, and A. Passito, ``Lightweight DDoS flooding attack detection using NOX/OpenFlow," in Proc. IEEE Local Comput. Netw. Conf., Oct. 2010, pp. 408415.

[10] S. A. Mehdi, J. Khalid, and S. A. Khayam, ``Revisiting traffic anomaly detection using software dened networking," in Proc. Int. Work- shop Recent Adv. Intrusion Detection. Berlin, Germany: Springer, 2011, pp. 161180.

[11] Y. Cui, L. Yan, S. Li, H. Xing, W. Pan, J. Zhu, and X. Zheng, ``SD-Anti- DDoS: Fast and efcient DDoS defense in software-dened networks," J. Netw. Comput. Appl., vol. 68, pp. 6579, Jun. 2016.

[12] H. Wang, L. Xu, and G. Gu, ``FloodGuard: A DoS attack prevention extension in software-dened networks," in Proc. 45th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw., Jun. 2015, pp. 239250.

.