DDoS Attack Detection Algorithms Based on Pattern Classification and Machine Learning

D.O.I - 10.51201/Jusst12593

http://doi.org/10.51201/Jusst12593

[1]Anup Ingle, [2]Dr. Avinash Gour, [3]Dr. Ketki Kshirsagar

[1]Research Scholar, Dept. of Electronics and Communication Engineering, Sri Satya Sai University of Technology & Medical Sciences, Sehore, Bhopal Indore Road, Madhya Pradesh, India
[2]Research Guide, Dept. of Electronics and Communication Engineering, Sri Satya Sai University of Technology & Medical Sciences, Sehore, Bhopal Indore Road, Madhya Pradesh, India
[3]Research Co-Guide, Dept. of Electronics and Telecommunication, VIIT, Pune, Maharashtra, India

Abstract:*Damage from DDoS attack in increasing day by day and an efficient attack detection algorithm is urgently needed. Many current DDoS algorithms are based on anomaly detections which are ineffective in real environment. Detection DDoS attack can be tackled effectively with pattern classification based on flow of packet and machine learning algorithms. In this paper three such pattern classificationsbased on flow of packet and machine learning based algorithm for detection of DDoS attack are discussed. Implementation of these algorithms gives better accuracy in limited time and memory space; hence it's one of the highly scalable and effective in detection of DDoS attack.*

Keywords: *DDoS, Flow-based Pattern Classification, Machine Learning, LSTM.*

1. Introduction

With the evolution of Internet, so do the threats over the Internet arises and severity of Distributed Daniel of Service (DDoS) attack is unquestionable. Damage from DDoS attack in increasing day by day and an efficient attack detection algorithm is urgently needed. Many current DDoS algorithms are based on anomaly detection or signatures based and hence not scalable and efficient. Anomaly detection algorithms are inefficient because of their restricted application over wide variety of DDoS attack. Anomaly detection based algorithms are easy to implement in codes but require huge changes to match up with the upcoming ways of DDoS attack. The challenge of detecting DDoS attack can be tackled effectively with pattern classification and machine learning algorithms/ model. Implementation of these algorithms gives better accuracy in limited time and memory space; hence it's one of the highly scalable and effective in detection of DDoS attack.

2. Detection algorithm for DDoS attack

There are many types of DDoS attacks but the most prominent are TCP SYN, UDP Flood and Ping Flood attacks. These types of attacks are mainly comprised of exploitation of inherently present loop holes in the associated protocols. For example TCP protocol always have SYN, SYN-ACK, ACK flags to completely establish TCP SYN channel but attacker only keeps on sending SYN signal but no ACK signal. Clearly DDoS attacks are easy to perform but very hard to detect in the local area network (LAN) environment for the legitimate user.

In our algorithms for detecting DDoS attack, we have used two different thread based design in order to detect DDoS attack within three packets. Memory and time efficiency is the primary concern for these algorithms. First thread deals with catching and matching the attack patterns with respective packets and the second thread deals with detection of DDoS attack. All the three algorithmsmentioned in succeeding section III, IV and Vare based on pattern classification using decision tree algorithm in detecting the attack in local area network. And section VI based on Long Short Term Memory (LSTM) machine learning model to detect if there is an attack in progress.

3. TCP-SYN Flood attack detection algorithm

Firstly TCP Packet are captured based on their protocol and two successive packet are observed by storing captured in Class TCP-SYN. Attributes of class TCP-SYN are Serial Number, Time Stamp (Arrival Time), Source Address 1 of packet n, Destination Address 1 of packet n, Source Address 2 of packet n+1, Destination Address 2 of packet n+1, Source Address 3 of packet n+2, Destination Address 3 of packet n+2.

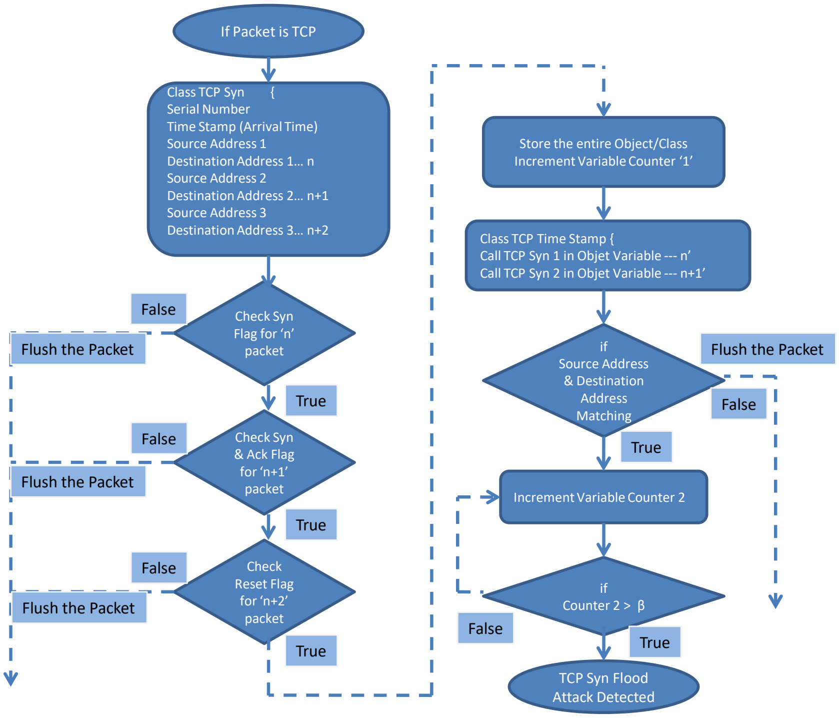
If Packet is TCP
Class TCP Syn {
Serial Number
Time Stamp (Arrival Time)
Source Address 1
Destination Address 1… n
Source Address 2
Destination Address 2… n+1
Source Address 3
Destination Address 3… n+2
Check Syn Flag for 'n' packet
False
Flush the Packet
True
Check Syn & Ack Flag for 'n+1' packet
False
Flush the Packet
True
Check Reset Flag for 'n+2' packet
False
Flush the Packet
True
Store the entire Object/Class Increment Variable Counter '1'
Class TCP Time Stamp {
Call TCP Syn 1 in Objet Variable --- n'
Call TCP Syn 2 in Objet Variable --- n+1'
if Source Address & Destination Address Matching
Flush the Packet
False
True
Increment Variable Counter 2
if Counter 2 > β
False
True
TCP Syn Flood Attack Detected

Figure 1: Flow-chart for Detection of TCP-SYN Flood Attack

For the 'n' packet check if protocol is TCP and SYN Flag, if false then flush the buffer because packet is not TCP type, but if the packet SYN flag is true then go to 'n+1' packet and check SYN& ACK Flag, if ACK is false then flush the buffer, but if SYN& ACKboth are true then go to 'n+2' packet and check Reset Flag if Reset flag is false then it's a "Legitimate Packet", no need to store it on the buffer. Now if the reset flag is True then store the entire Object/Class in the declared buffer for further investigation and increment a Variable called Counter (Counter ++).

Now in order to check DDoS attack pattern we use the stored buffer on non-legitimate packet for TCP. A separate thread is invoked for class called TCP Time Stamp. Attributes of this class are previous class TCP-SYN 1 in Objet Variable for n packet, similarly TCP-SYN 2 in Objet Variable for n+1. For matching the pattern we will check if Source Address & Destination Address is matching in TCP-SYN 1 & TCP-SYN 2 if not then flush the entire buffer, but if true then increment the variable called Counter 2 in TCP Time Stamp class (Counter ++). To declare it's a TCP-SYN attack check Counter is greater than 100. This value of counter can be varied depending upon the requirement of user to any number greater than 10.If the counter variable is 'True' then print "TCP-SYN Flood Attack Detected" with necessary action to check the attack.

4. UDP Flood attack detection algorithm

Firstly UDP Packets are captured based on their protocol and two successive packets are observed by storing captured in Class UDP Attack. Attributes of class UDP Attack are Serial Number, Time Stamp (Arrival Time), Source Address 1 of packet n, Destination Address 1 of packet n, Source Address 2 of packet n+1, Destination Address 2 of packet n+1, Source Address 3 of packet n+2, Destination Address 3 of packet n+2.

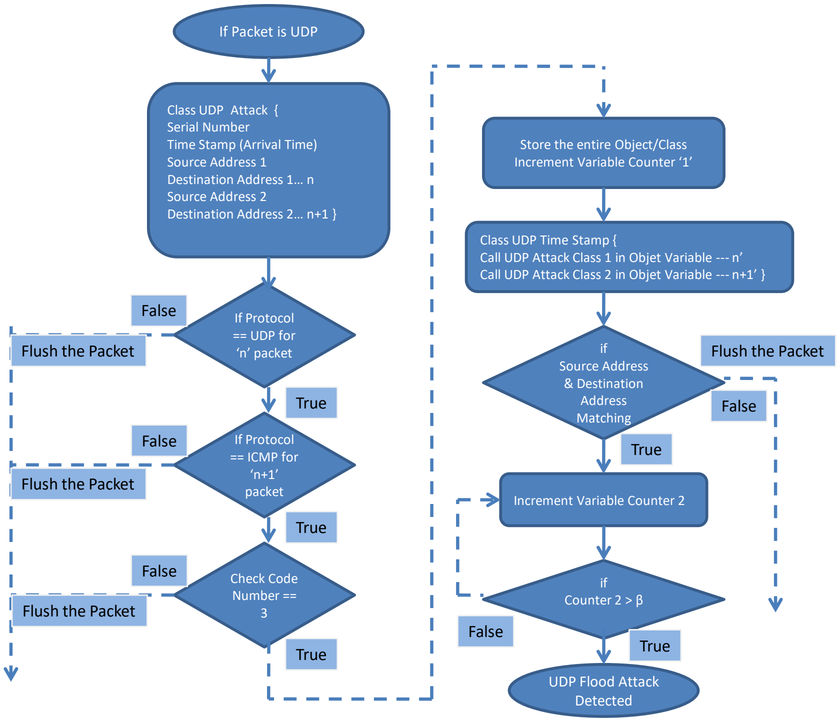
If Packet is UDP
Class UDP Attack {
Serial Number
Time Stamp (Arrival Time)
Source Address 1
Destination Address 1… n
Source Address 2
Destination Address 2… n+1 }
False
If Protocol == UDP for 'n' packet
Flush the Packet
True
False
If Protocol == ICMP for 'n+1' packet
Flush the Packet
True
False
Check Code Number == 3
Flush the Packet
True
Store the entire Object/Class
Increment Variable Counter '1'
Class UDP Time Stamp {
Call UDP Attack Class 1 in Objet Variable --- n'
Call UDP Attack Class 2 in Objet Variable --- n+1' }
if Source Address & Destination Address Matching
Flush the Packet
False
True
Increment Variable Counter 2
if Counter 2 > β
False
True
UDP Flood Attack Detected

Figure 2: Flow-chart for Detection of UDP Flood Attack

For the 'n' packet check protocol is UDP, if false then flush the buffer because packet is not UDP type, but if the packet has UDP protocol true then go to 'n+1' packet and check if the protocol is ICMP, if not ICMP then flush the buffer it's a legitimate packet and need not required to buffer, but if ICMP protocol is there go to 'n+2' packet and check Code Number ==3(Destination Port Unreachable) if false then it's a "Legitimate Packet", no need to store it on the buffer but if True then store the entire Object/Class in the declared buffer for further investigation and increment a Variable called Counter (Counter ++).

Now in order to check DDoS attack pattern we use the stored buffer on non-legitimate packet for UDP. A separate thread is invoked for class called UDP Time Stamp. Attributes of this class are previous class UDP Attack 1 in Objet Variable for n packet, similarly UDP Attack 2 in Objet Variable for n+1. For matching the pattern we will check if Source Address & Destination Address is matching in UDP Attack 1 & UDP Attack 2 if not then flush the entire buffer, but if true then increment the variable called Counter in UDP Time Stamp class (Counter ++). To declare it's a UDP Flood attack check Counter is greater than 100. This value of counter can be varied depending upon the requirement of user to any number greater than 10.If the counter variable is 'True' then print "UDP Flood Attack Detected" with necessary action to check the attack.

5. ICMP Flood attack detection algorithm

Firstly ICMP Packets are captured based on their protocol and two successive packets are observed by storing captured in Class ICMP Attack. Attributes of class ICMP Attack are Serial Number, Time Stamp (Arrival Time), Source Address 1 of packet n, Destination Address 1 of packet n, Source Address 2 of packet n+1, Destination Address 2 of packet n+1, Source Address 3 of packet n+2, Destination Address 3 of packet n+2.

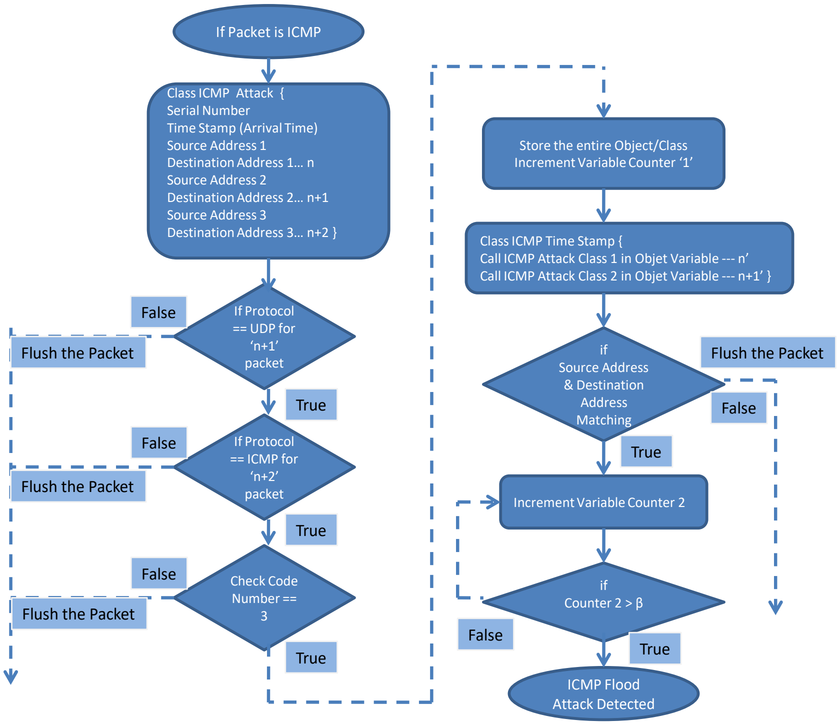
If Packet is ICMP
Class ICMP Attack {
Serial Number
Time Stamp (Arrival Time)
Source Address 1
Destination Address 1… n
Source Address 2
Destination Address 2… n+1
Source Address 3
Destination Address 3… n+2 }
False
Flush the Packet
If Protocol == UDP for 'n+1' packet
True
False
Flush the Packet
If Protocol == ICMP for 'n+2' packet
True
False
Flush the Packet
Check Code Number == 3
True
Store the entire Object/Class
Increment Variable Counter '1'
Class ICMP Time Stamp {
Call ICMP Attack Class 1 in Objet Variable --- n'
Call ICMP Attack Class 2 in Objet Variable --- n+1' }
if Source Address & Destination Address Matching
Flush the Packet
False
True
Increment Variable Counter 2
if Counter 2 > β
False
True
ICMP Flood Attack Detected

Figure 3: Flow-chart for Detection of ICMP Flood Attack

For the 'n' packet check protocol is ICMP, if false then flush the buffer because packet is not ICMP type, but if the packet has ICMP protocol true then store Type and Code and go to 'n+1' packet and check if the protocol is UDP and check till the protocol is UDP, if the protocol is UDP go to 'n+2' packet check if protocol is ICMP and then check Code Number==3(Destination Port Unreachable) if false then it's a "Legitimate Packet", no need to store it on the buffer but if True then store the entire Object/Class in the declared buffer for further investigation and increment a Variable called Counter (Counter ++).

Now in order to check DDoS attack pattern we use the stored buffer on non-legitimate packet for ICMP. A separate thread is invoked for class called ICMP Time Stamp. Attributes of this class are previous class ICMP Attack 1 in Objet Variable for n packet, similarly ICMP Attack 2 in Objet Variable for n+1. For matching the pattern we will check if Source Address & Destination Address is matching in ICMP Attack 1 & ICMP Attack 2 if not then flush the entire buffer, but if true then increment the variable called Counter in ICMP Time Stamp class (Counter ++). To declare it's an ICMP Flood attack check Counter is greater than 100. This value of counter can be varied depending upon the requirement of user to any number greater than 10.If the counter variable is 'True' then print "ICMP Flood Attack Detected" with necessary action to check the attack.

6. Machine Learning:

- **Detecting SYN Flood attacks using Machine Learning**

  The first step was to compare and analyze various existing solutions decision trees, support vector machines, clustering algorithms, etc.

  All these algorithms analyze each packet and then classify it into one of two classes (normal, attack). When in reality, there is also a time component necessary to consider when detecting whether a DOS attack is in progress. For this reason, we decided and implemented a Long Short Term Memory (LSTM) machine learning model to detect if there is an attack in progress.

- **Acquiring data**

  The first step in developing a model for DOS detection was downloading a suitable dataset consisting records of the required metadata extracted for the packets. This dataset should be labeled and consist of two continuous streams of packet collected during normal and attack conditions. The dataset we used was about 700 MB in CSV format and consisted of 56,58,998 packets.

- **Preprocessing data**

  The raw data obtained from the dataset needs to be pre-processed before training. This includes removing unnecessary data like source and destination IP, and protocol from the dataset. We then standardize the data in the data columns and convert them into appropriate data types so as to not waste space on the main memory. We also create labels where 1 signifies attack and 0 signifies normal condition. This data is then separated into training and testing data.

- **Computational requirements**

  The entire dataset when pre-processed occupies about 130 GBs of space in the main memory (RAM). Small batches of this data are fed to the model for training. The model consists of one bidirectional and two dense network layers and this contains 62,721 trainable parameters, therefore the computational requirements are significant.

- **The model**

  The model is a Keras sequential model with 3 layers (Bidirectional LSTM, Dense [128], and Dense [1]). We use tanh, relu and sigmoid activation for the layers, respectively. It has 62,721 trainable parameters, most of which belong to the LSTM layer. We have used 'L2' regularization for regularizing the weights in our model. The model when trained for 120 epochs with a batch size of 32 yielded training accuracy of 99.08% and testing accuracy of 98.78% which is a significant improvement over other solutions. The precision and recall are 98.39% and 99.80%

which are pretty well balanced. The result shown in figure 4 is the plot of accuracy verses epoch and figure 5 is the plot of running precision verses epoch.

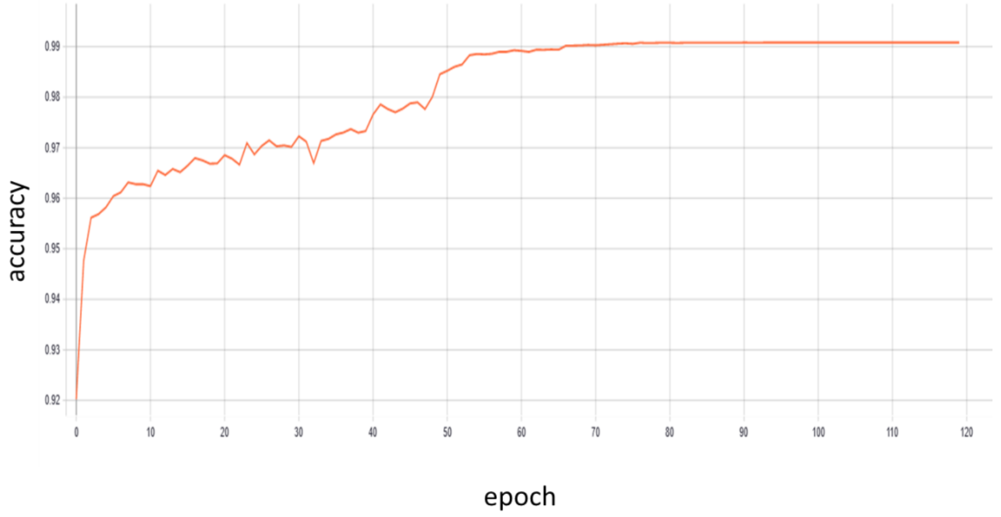

Figure 4: Plot of accuracy vs epoch

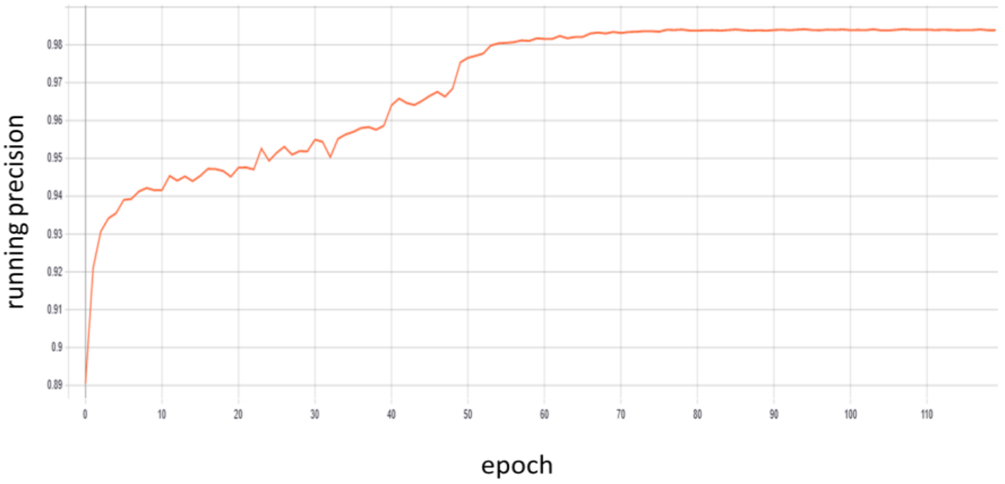

Figure 5: Plot of running precision vs epoch

7. Conclusion

By using pattern classification based of flow of packet and machine learning model, a more scalable and effective way of detecting DDoS attack is obtained. This algorithm is easy to implement and manage for new trends trend in DDoS attack. The memory requirement for these algorithms are small since it is based on pattern classification not on actual signature or anomaly based algorithm where data needed to be stored in the database. Hence Algorithms are zero database algorithm and data mining overheads ore completely omitted once the model is trained and deployed in the system as application specific integrated service. Because of algorithms time response attributes which is very less, these algorithms can be used or implemented for real time or mission critical applications. It can be also implemented in the edge router or border router with dedicated trained model incorporate within the hardware itself.

8. References

[1] *B. Thangaparvathi, D. Anandhavalli and S. M. Shalinie, "A high speed decision tree classifier algorithm for huge dataset," 2011 International Conference on Recent Trends in Information Technology (ICRTIT), Chennai, Tamil Nadu, 2011, pp. 695-700, doi: 10.1109/ICRTIT.2011.5972267.*

[2] *Y. Ohsita, S. Ata and M. Murata, "Detecting distributed denial-of-service attacks by analyzing TCP SYN packets statistically," IEEE Global Telecommunications Conference, 2004. GLOBECOM '04., Dallas, TX, 2004, pp. 2043-2049 Vol.4, doi: 10.1109/GLOCOM.2004.1378371.*

[3] *Haris, S.H.C. & Ahmad, R.Badlishah&Abd Ghani, Mohd. (2010). Detecting TCP SYN flood attack based on anomaly detection. Network Applications, Protocols and Services, International Conference on. 240-244. 10.1109/NETAPPS.2010.50.*

[4] *Huan-Rong Tang, Rou-Ling Sun and Wei-Qiang Kong, "Wireless Intrusion Detection for defending against TCP SYN flooding attack and man-in-the-middle attack," 2009 International Conference on Machine Learning and Cybernetics, Hebei, 2009, pp. 1464-1470, doi: 10.1109/ICMLC.2009.5212317.*

[5] *Cuihong Wu, "The problems in campus network information security and its solutions," 2010 2nd International Conference on Industrial and Information Systems, Dalian, 2010, pp. 261-264, doi: 10.1109/INDUSIS.2010.5565862.*

[6] *M. A. Rahman and E. Al-Shaer, "A declarative approach for global network security configuration verification and evaluation," 12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops, Dublin, 2011, pp. 531-538, doi: 10.1109/INM.2011.5990556.*

[7] *S. M. Shalinie et al., "CoDe — An collaborative detection algorithm for DDoS attacks," 2011 International Conference on Recent Trends in Information Technology (ICRTIT), Chennai, Tamil Nadu, 2011, pp. 113-118, doi: 10.1109/ICRTIT.2011.5972338.*

[8] *S. Kumar, M. Azad, O. Gomez and R. Valdez, "Can Microsoft's Service Pack2 (SP2) Security Software Prevent SMURF Attacks?," Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services (AICT-ICIW'06), Guadelope, French Caribbean, 2006, pp. 89-89, doi: 10.1109/AICT-ICIW.2006.60..*

[9] *Z. Wang and X. Wang, "DDoS attack detection algorithm based on the correlation of IP address analysis," 2011 International Conference on Electrical and Control Engineering, Yichang, 2011, pp. 2951-2954, doi: 10.1109/ICECENG.2011.6057035.*

[10] *V. A. Siris and F. Papagalou, "Application of anomaly detection algorithms for detecting SYN flooding attacks," IEEE Global Telecommunications Conference, 2004. GLOBECOM '04., Dallas, TX, 2004, pp. 2050-2054 Vol.4, doi: 10.1109/GLOCOM.2004.1378372.*

[11] *A. Yamada, Y. Miyake, K. Takemori and T. Tanaka, "Intrusion detection system to detect variant attacks using learning algorithms with automatic generation of training data," International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume II, Las Vegas, NV, 2005, pp. 650-655 Vol. 1, doi: 10.1109/ITCC.2005.178.*

[12] *H. Hamed and E. Al-Shaer, "Taxonomy of conflicts in network security policies," in IEEE Communications Magazine, vol. 44, no. 3, pp. 134-141, March 2006, doi: 10.1109/MCOM.2006.1607877.*

[13] *K. Gkoutioudi and H. Karatza, "A Simulation Study of Multi-criteria Scheduling in Grid Based on Genetic Algorithms," in 2012 IEEE 10th International Symposium on Parallel and Distributed Processing with Applications (ISPA), Leganes, 2012 pp. 317-324.doi: 10.1109/ISPA.2012.48.*

[14] *Q. Meng, Q. He, N. Li, X. Du and L. Su, "Crisp Decision Tree Induction Based on Fuzzy Decision Tree Algorithm," 2009 First International Conference on Information Science and Engineering, Nanjing, 2009, pp. 4811-4814, doi: 10.1109/ICISE.2009.440.*

[15] *Z. Liu and X. Zhang, "Prediction and Analysis for Students' Marks Based on Decision Tree Algorithm," 2010 Third International Conference on Intelligent Networks and Intelligent Systems, Shenyang, 2010, pp. 338-341, doi: 10.1109/ICINIS.2010.59.*

[16] *A. Navada, A. N. Ansari, S. Patil and B. A. Sonkamble, "Overview of use of decision tree algorithms in machine learning," 2011 IEEE Control and System Graduate Research Colloquium, Shah Alam, 2011, pp. 37-42, doi: 10.1109/ICSGRC.2011.5991826.*

[17] *Alsadhan, Abeer& Hussain, Abir& Alani, Mohammed. (2018). Detecting NDP Distributed Denial of Service Attacks Using Machine Learning Algorithm Based on Flow-Based Representation. 134-140. 10.1109/DeSE.2018.00028.*

[18] *Francisco Sales de Lima Filho, Frederico A. F. Silveira, Agostinho de Medeiros Brito Junior, Genoveva Vargas-Solar, Luiz F. Silveira, "Smart Detection: An Online Approach for DoS/DDoS Attack Detection Using Machine Learning", Security and Communication Networks, vol. 2019, Article ID 1574749, 15 pages, 2019. https://doi.org/10.1155/2019/1574749.*

[19] *ZHU, MIN-JIE & GUO, NAI-WANG. (2018). Abnormal Network Traffic Detection Based on Semi-Supervised Machine Learning. DEStech Transactions on Engineering and Technology Research. 10.12783/dtetr/ecame2017/18466.*