

Anomaly detection for software defined datacenter networks using X-Pack

Sneha Mahabaleshwar¹, Shobha Gangadhar¹, Sharath Krishnamurthy²

¹Department of Computer Science and Engineering, RV College of Engineering, Bengaluru, India

²Principal Technical Specialist, NOKIA Solutions and Networks India Pvt. Ltd., Bengaluru, India

Article Info

Article history:

Received Oct 27, 2022

Revised Mar 31, 2023

Accepted Apr 2, 2023

Keywords:

Anomaly detection

DoS attack

Elasticsearch

Kibana

Machine learning

Software defined networks

ABSTRACT

The global data center market is growing as more and more enterprises are increasingly adopting cloud computing services and applications. Data centers are evolving towards highly virtualized architectures where transformation to software defined network (SDN) based solutions provides benefits in terms of network programmability, automation, and flow visibility. With the benefits, the need for securing network becomes essential as many critical applications are hosted on to such networking platforms. Anomaly detection is a continuous process of monitoring the traffic pattern and alerting the user about the anomalies if detected. For such real time analysis NoSQL and relational databases are less efficient. This paper proposes a framework for anomaly detection and alerting system using Elasticsearch database for SDN. Traffic patterns generated from SDN devices are continuously monitored and predefined actions are taken immediately if an anomaly is detected. The proof of concept is implemented in NOKIA's Nuage Networks Laboratory and the results showed a real time anomaly detection and took relevant actions within minimum time.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Sneha Mahabaleshwar

Department of Computer Science and Engineering, RV College of Engineering

Bengaluru, Karnataka, 560060, India

Email: sneham@rvce.edu.in

1. INTRODUCTION

The cyber-attacks are increasingly becoming common and harder to detect, causing significant damage in terms of time and money by causing wide spread disruption to key services. Li and Liu [1] have done a detailed report on the basic definitions and concepts of cyberspace, sources of cyber threats, different types of cyber attacks, different methods used by cyber criminals. They also highlight the need of security detection and prevention systems not only in government sectors but also in the private sectors. One of the methods to solve this issue is to use machine learning (ML) [2] based techniques to detect anomalies in the network by looking at patterns in traffic flow data in near real time, and enable automated response to mitigate the risks associated with these attacks. Since the main aim of this research is to develop a framework for detecting anomalies in the traffic pattern using machine learning, review of literature focuses on such efforts in finding different types of attacks like denial of service (DoS) or distributed denial of service attacks (DDoS).

Suresh and Anitha [3] have considered datasets from public-domain CAIDA dataset. The traffic was collected on the smart and secure environment (SSE) network for their study which include various DDoS attack patterns in them. They have done a comparison analysis between Fuzzy C Means, K-nearest neighbor (KNN), support vector machines (SVM), decision tree (DT), K-Means, six ML algorithms and with experiments proved that fuzzy C-Means clustering gives the better classification and is fast among all the other algorithms. Saini *et al.* [4] have used a machine learning tool called WEKA for detecting and defining DDoS

attacks. Using four different classifiers J48, MLP, random forest, Naive Bayes, comparative analysis was done based on maximum identification of true positive cases. With accuracy calculation of precision and recall was also done to prove that J48 outperformed the rest of the algorithms. Shaukat *et al.* [5] have done a very detailed survey on the use of ML algorithms and cyber security in last decade. They have outlined all the ML algorithms used till now for cyber security which have proved significant results.

Shetty *et al.* [6] have compared the accuracy of detection using neural networks, principal component analysis, decision tree, random forest and SVM on KDD99 intrusion dataset. But the authors have tested the algorithm on the collected dataset rather than on the real time dataset. Many applications which use cloud computing have seen tremendous growth in the recent years. This made many organizations to migrate their applications to cloud datacenters. This was made easy with the wild adoption of software defined networks (SDN). With the advantages of SDN there were added security risks. Significant efforts are being made to mitigate the security risks involved in adopting SDN.

El-Shamy *et al.* [7] have considered SVM to develop a SDN based monitoring system which identified the performance anomaly and identified the issues of the distributed application. One class support vector machines and multiclass support vector machines have been used to monitor the anomalous behavior in the performance of different applications. Jafarian *et al.* [8] have done a detailed survey on different many anomaly detection algorithms in SDN. Their study was categorized into different anomaly detection schemes. In the authors view Denial of Service attacks are considered as the most significant external threats in SDNs. Abdelrahman *et al.* [9] reviewed the security solutions for the vulnerabilities of the current controllers and the countermeasures.

Farhana *et al.* [10] proposed an intrusion detection system using deep neural networks for anomaly detection in computer network patterns. CICIDS2017 dataset was used for training and testing which contains real network traffic data. The authors were able to get 99.13% accuracy for binary classification and 99.29% for multi-class classification on the said dataset. But the model failed to classify few of the data as the number of records were insufficient. Al-Shabi and Abuhamdah [11] proposed a model which used adapted long shortterm memory (LSTM) recurrent neural networks (RNN) for abnormal behavior in IoT systems. Intel labs dataset was used for evaluating the proposed model. Negoita and Carabas [12] proposed a model which uses machine learning algorithms for attack detection and monitoring using built-in machine learning jobs of Elasticsearch to enhance the security of an application and infrastructure of the cloud datacenter. Monitoring was done using snort and network analytics was collected by Metricbeat and Packetbeat. The jobs were created through Kibana on the Elasticsearch indices to monitor the traffic. This framework is very close to the method proposed in this paper for developing a framework using Elasticsearch ML APIs. Folino *et al.* [13] have developed a framework using ELK stack to monitor user behaviour for and detect anomalies in real time using the Kubernetes platform. Hariharan *et al.* [14] came up with CAMLPAD where they have used anomaly detection technique using scoring method like Elasticsearch X-Pack.

All these works give the significance of using machine learning algorithm for anomaly detection in various network setups. However, many of the implementations discussed have considered static dataset which may or may not work in the real time traffic monitoring. There is a requirement to detect the anomalies in the traffic as quick as possible to take proper countermeasures, as the attack may not be always longer enough. To develop such a framework to detect DoS kind of attacks, a case study has been conducted in NOKIAs Nuage Networks laboratory so that the proposed solution is well suited for any industry which offers SDN based solution with added security feature.

NOKIAs Nuage Network offers different solution with which it offers a security service called virtualized security services (VSS) [15]. VSS is a software-defined security solution that is based on Nuage virtualised services platform (VSP) to help address protection, detection and operational challenges in cloud, data center, and branch environments. VSS is a distributed, end-to-end (cloud, DC, branch) software-defined network security, visibility, and security automation solution. VSS has capabilities that provides traffic visibility, security monitoring and immediate incident response. The ability to address the security challenges is end-to-end (branch, data center, and cloud), across heterogeneous workload types (VM: multi hypervisor, containers, bare-metal) and on any existing IP underlay network.

Currently VSS uses threshold crossing alerts (TCA) feature, based on traffic metrics like Packets in, Packets out, Bytes in, Bytes out, anterior cruciate ligament (ACL) deny event count, anti-spoof event count, that could be because of a DoS [16]-[18] attack, for generating real time alerts. TCA uses static threshold calculation to decide on the anomalies in the traffic pattern. This system was not efficient in finding out all types of anomalous behavior in the traffic patterns and raise an alarm back to the user. There is a need for dynamic calculation based on the traffic observed for the threshold to have proper alarms been generated. X-Pack [19] is an elastic stack extension that provides many capabilities like traffic pattern monitoring, analyzing, and reporting. Using these APIs a framework can be developed to monitor the network in real-time for detecting unusual behavior and take relevant actions.

Since Nuage Network uses Elasticsearch as its back-end database where all the flow statistics are getting collected. Review was conducted on Elasticsearch based anomaly detection to leverage the usage of existing techniques for developing the proposed model. Komisarek *et al.* [20] developed tool for implementing machine learning algorithms for anomaly detection and cyber-attack detection in Network traffic data. They have used Apache Kafka for communication between applications. Elasticsearch and Kibana for storage and visualization of traffic data. Shah *et al.* [21] have developed a framework for analyzing twitter data using Elasticsearch and Kibana.

Contributions: After going through the existing system and the literature review for coming up with a solution to the problem Nuage Network was facing in their VSS product, the following design strategies are made which are the main contributions of this paper:

- a) Designing and developing a framework for VSS that dynamically calculates the threshold to detect anomalies in the traffic pattern using X-Pack Elasticsearch ML API.
- b) Experimentally proving the effectiveness of the developed feature in NOKIA's Nuage Networks laboratory, to detect the anomalies which TCA could not detect.
- c) Take necessary actions upon detecting the anomalies based on the policies defined in VSS.

2. PROPOSED FRAMEWORK USING ELASTICSEARCH X-PACK

2.1. Existing system: virtualized security services (VSS) and threshold crossing alerts (TCA)

Nuage networks offers different solutions such as; virtualized network services (VNS) which delivers SD-WAN, virtualized cloud services (VCS) is SDN solution for telco cloud data center infrastructure, SD-security which uses the power of software-defined networking (SDN and SD-WAN) to prevent, detect and respond to security incidents across the WAN, datacenter and cloud [15]. VSS is a Nuage Solution offered for both SDN and SD-WAN as explained in the introduction section. Nuage uses Elasticsearch as its back-end database for all its solutions. Elasticsearch [22] is a full text java-based search engine, specifically designed keeping cloud environment in mind. It helps in solving main issues of scalability, search in real time, and efficiency that relational databases fail to address. Apart from these, Elasticsearch can also be integrated with big data tools, gives structured search capability with static analysis and real-time monitoring which can be visualized using Kibana. Being a distributed data storage system Elasticsearch can store and fetch documents in real-time which are serialized as JSON [23]. Each document is stored in to an index on to Elasticsearch and are searched using JSON objects upon which Elasticsearch returns the results also in JSON format. By default, Elasticsearch is clustered [24]. No special configurations are required, they hook up with each other to become a cluster.

The existing framework of VSS with TCA configured is as shown in Figure 1. In the data plane there are switching and forwarding specific solutions called as network services gateway (NSG) and virtual routing and switching (VRS), at the control plane virtual services controller (VSC) and at the management plane virtual services directory (VSD). The NSG/VRS uses Openflow protocol to communicate with VSC and VSC uses the XMPP protocol to communicate with VSD. All the flow information is regularly stored on to the Elasticsearch database in to different indices (flow/vport) at a regular interval of time. These flows get enriched with additional information, specific to the network, from VSD before it is stored on to Elasticsearch.

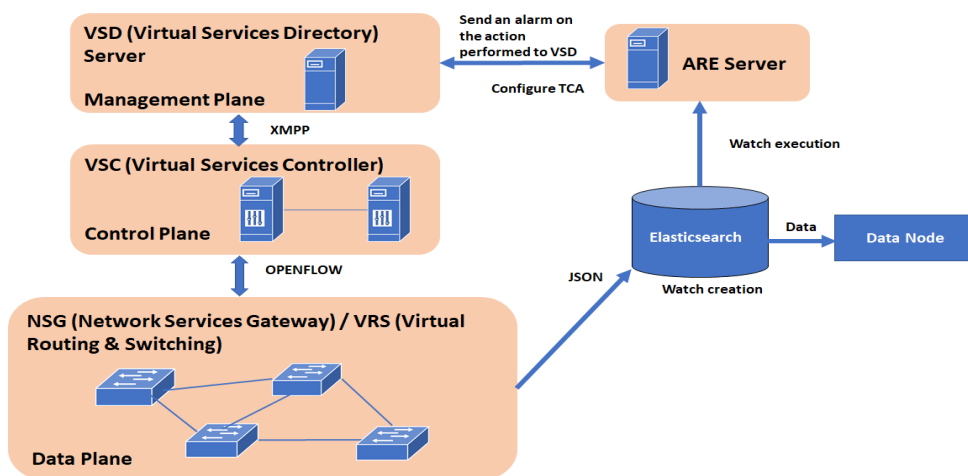


Figure 1. Existing framework of VSS with TCA creation and alarm generation

TCA's can be created through alerting and response engine (ARE) server on different levels to monitor the network. Example TCA creation is shown in Figure 2, where BytesIn metric is used. There are two parts in TCA creation; one is general description and the second is Alarm Triggering. TCA name, the metric on which TCA is being created, Throttle Time and action type is selected in the General Description part. Type of traffic (mean, max, min, average), Threshold in bytes (90,000 Bytes) and Period in seconds (10 Sec) is set as part of the Alarm Triggering. Once the TCA for BytesIn metric is configured through the VSD, corresponding watch (3) is created. Every 10 seconds once the flow index is queried to feed the watcher input with the average BytesIn data which checks if the average BytesIn exceeds 90,000 bytes. If it exceeds then the condition met status of the watcher will be set to true and the selected action will be performed. All the communication from VSD to Elasticsearch happens through ARE server.

The limitation of the current system is even if the BytesIn reaches to 89,999 bytes or suddenly drops, then TCA will not generate any alarm because the status of the watcher will not return true. Gradual increase in BytesIn over time above the static threshold will result in too many alerts unless the static threshold is reconfigured. Dynamic threshold is needed so that all types of variation (sudden high/sudden low spikes) are caught.

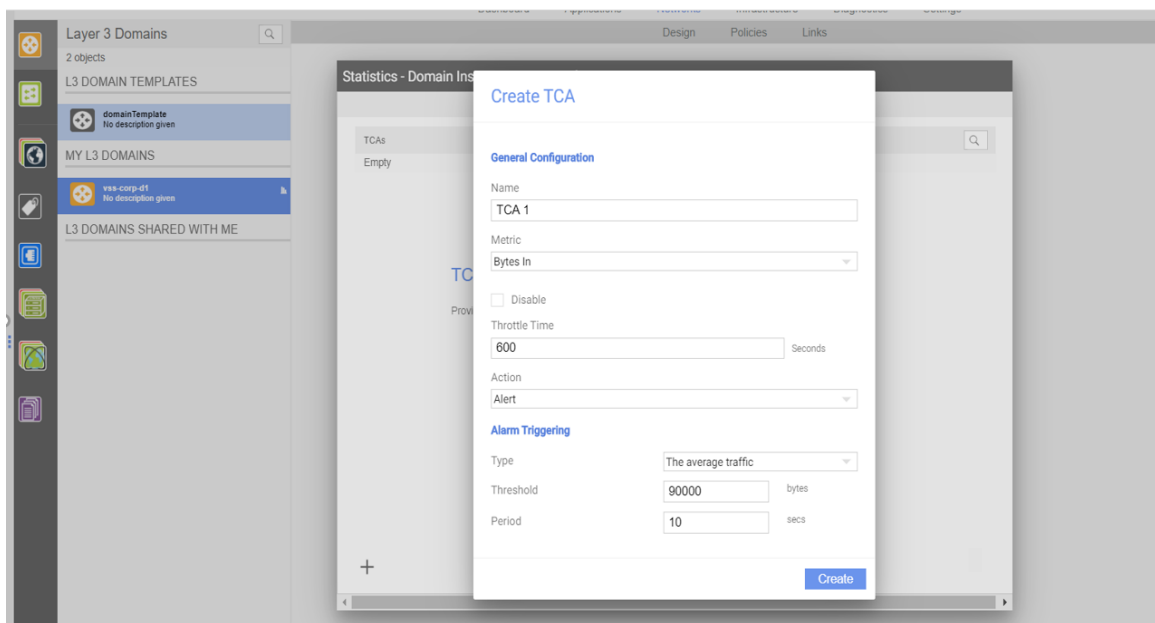


Figure 2. Screenshot for TCA creation for BytesIn metric from VSD user interface

2.2. VSS with Elasticsearch X-PACK

After going through the literature it's evident that the usage of ML algorithms is better in calculating the threshold dynamically for generating alerts through TCA rather than having a static threshold. Since Nuage Network uses Elasticsearch as its database, it is more relevant to use the ML model defined by Elastic stack to do this change. This choice decreases the time to feed the ML model with the data from the respective indices and increases the performance. The reasons for making this choice are:

- Elasticsearch gives an ML extension X-Pack with it, which can be used to create ML jobs to continuously monitor the traffic.
- And these jobs are designed to run on the Elasticsearch on a separate node than the data node but in the same cluster.

2.2.1. Proposed modification to VSS

Significant changes had to be made to the existing architecture to include ML capabilities. Figure 3 shows the modifications to the existing system. As shown in Figure 3 instead of creating a TCA with a static threshold ML job is created through the ARE server with the configuration specified by the user through the VSD UI and a corresponding watcher is also created to keep track of the changes in the specified parameter in the job. The watch is fired when there is an unusual change in the parameter for which this watch has been created. The actions specified in the watcher definition is executed and the alert is sent back to the VSD UI.

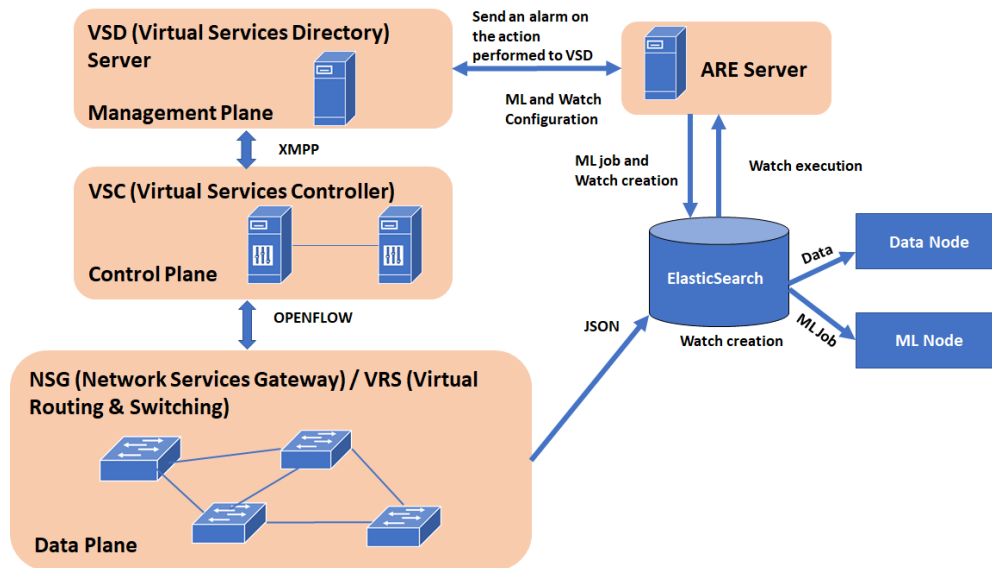


Figure 3. Overall architecture of the proposed system

2.2.2. Anomaly detection jobs

ML jobs can be created based on the number of detectors or can be custom built by the users. In this paper two types of ML jobs are being experimented; Single metric jobs and Multi-metric Jobs. Every ML job thus created has certain required fields: *analysis_config* and *data_description* with optional other fields. Single metric job has a single detector where as multi-metric job has more than one detector. The *analysis_config* includes *bucket_span*, *array_of_detectors*, *influencers*, *latency*, *per_partition_categorization*, *multivariate_by_fields*, *model_prune_window*, *summary_count_field_name*, the description of these fields can be found in [25]. A *detector* is the one which applies an analytical function to selected fields in data supplied. Different properties of detectors like, *by_field*, *over_field name*, *partition_field_name*, *by_field_name* can be set to define how the data should be viewed in the job during the analysis for detecting unusual behaviour. For example, the *by field* is set, then the field is used to split the data and use it for analyzing with respect to their own history, so that unusual values are found in the context of the split. Different examples on creating detectors are listed in. The job can also have a list of *influencers* that are responsible for the detected anomalies, in a way someone or something that influences or contributes to the anomalies in the data.

The default time interval for collecting the data from different indices for the ML jobs through datafeed is 10sec and the default data retention period for the models is 7days. **.ml-anomalies-shared* is the default location for all machine learning jobs to dump the detailed analysis results. Or there is also an option for creating a custom index.

2.2.3. Datafeeds

Datafeeds are used to retrieve data from Elasticsearch indices for anomaly detection jobs. There is a one-to-one relationship between datafeed and anomaly detection job. The datafeed contains a query that runs at regular interval of time to feed the ML model created by the ML job. The required fields of Datafeeds are *indices*, an array of index names to fetch data from, *job_id* the unique identifier for the associated ML job already created, and an optional *query*. If the query is unspecified, then by default all records of the index are selected for the analysis. ML Job must be created before the datafeed.

2.2.4. Watcher

A watch definition has *trigger*, *input*, *conditions*, *actions* [26]. The *trigger* refers to time interval when the watch is triggered. The *input* has the query to fetch data from the indices. Whenever the *condition* is met, the watcher executes the actions listed in the watch definition. The *actions* are performed and the user is also alerted on the performed action which is reflected in the VSD UI.

2.2.5. ML algorithm behind X-Pack and anomaly scoring

Since X-Pack is a proprietary solution, the exact model is not revealed, but it is a combination of different machine learning algorithms which automates the anomaly detection even for the users who are not experts in ML. By observing the historical behaviour, unsupervised machine learning approach is used by X-

Pack ML to statistically model the time series data. Whenever ML job is created, a dynamic model is built and stored till the job is deleted. As the real-time data is being analysed the model becomes more matured and is assessed for unusualness level with respect to the previous experience. If the data's behaviour is being within the low probability ranges, then an anomaly record is created and stored in the **.ml-anomalies-shared* index with an anomaly score proportional to the probability. This score is normalized on a dynamic scale between 0 and 100, where 100 is the most unusual thing ever detected for the data set. Thus, one can integrate ML with Alerting so that they can get an alert based on desired severity of the anomaly score. Basically, the algorithm is a mixture of many algorithms including: clustering, time series decomposition, Bayesian distribution modelling and correlation analysis. The beauty of X-Pack is, it hides all the complexity of the hybrid model thus created by exposing only the required APIs to the users, using which they can program the model for their needs. The criticality of the anomaly scores given for each record are as mentioned in Table 1.

Table 1. Anomaly scoring

Criticality	Range of Anomaly Score
Warning	0 and above
Minor	25 and above
Major	50 and above
Critical	75 and above

3. METHOD

Since the model is hidden, the effectiveness of the developed framework must be proved by experimentation with different types of attack datasets. So, a proof of concept is implemented in NOKIA Nuage Network Laboratory, Bengaluru, India. Table 2 specifies the objects and their values being used in the system implementation. In Elasticsearch cluster by default, any node with X-Pack installed will be able to run machine learning jobs. But the minimum requirement in addition to memory utilization of Elasticsearch for running ML jobs would be 64GB RAM with 4 Cores. This configuration allows around 10 concurrent ML jobs. So more the number of ML nodes created, a greater number of ML jobs can be run concurrently.

Table 2. Default object values

Object	Field Name	Description	Default Value
ML analyzer	bucketSpan	The length of each bucket for ML job analysis, in seconds	300
	resultsRetention Days	The retention time in no. of days for the results of the analysis	7
	modelMemory Limit	The amount of memory in MB required for ML job analysis	1024
ML datafeed	frequency	The frequency with which queries are made while the datafeed runs in real time, in seconds	60
	queryDelay	The number of seconds behind real time that data is available for querying	30
	scrollSize	The size parameter for search results used in ES queries	100
EsWatch	trigger Interval	The interval with which the watch gets triggered, in seconds	30
	throttle Period	The throttle period for the watch action to be executed, in seconds	10
	threshold	The threshold for the anomaly score [0-100] for which action must be initiated	15
	period	The period in seconds to look back into the anomalies index from present time to evaluate watch condition	30

3.1. Execution flow of the proposed model

The topology considered for developing the proof of concept is shown in Figure 4. A VRS simulator is used to simulate the traffic between the hosts connected to switches S1 and S2. For every 10s, the packets are sent across the network by randomly choosing the source and the destination hosts. The packets thus sent are forwarded to Elasticsearch indices *vport_metric* and *flow_gen*. Two ML jobs are created; one for the *vport* and one for the TCP flags.

There are different ways of creating ML jobs in Elasticsearch; single metric jobs and multimetric jobs. In single metric jobs only one detector object can be created which can monitor only one metric, whereas in multimetric job multiple detectors can be created together in one job to monitor multiple metrics. Multimetric jobs are helpful if the input is same for all the detectors.

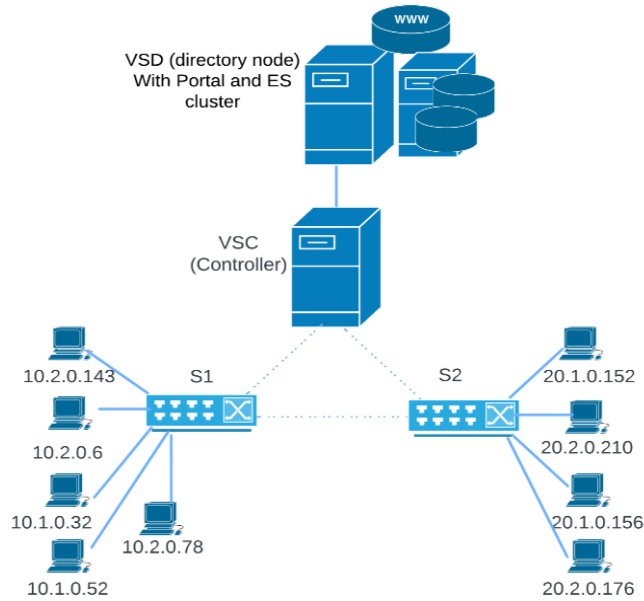


Figure 4. Topology used for proof of concept

Figure 5 shows a sample single metric analyzer creation for monitoring a vportId in a network on the BytesIn metric. The flowchart for creation of analyzer is depicted in the Figures 5(a) and 5(b) shows how the vportId ML analyzer can be created through VSD UI. The user must select the analyzer type, and associate the corresponding anomaly detector group, ML datafeed and ES watch, and create the analyzer. The user has been given an option for enabling/disabling the ML analyzer according to their requirements. After creating the job, the Bytes_in ML job is opened and the datafeed is started, which provides the data to the detector. The ML job analyzes the data in real time obtained from the datafeed and continues to store results of the analysis by default in the *.ml-anomalies-shared index with an anomaly score.

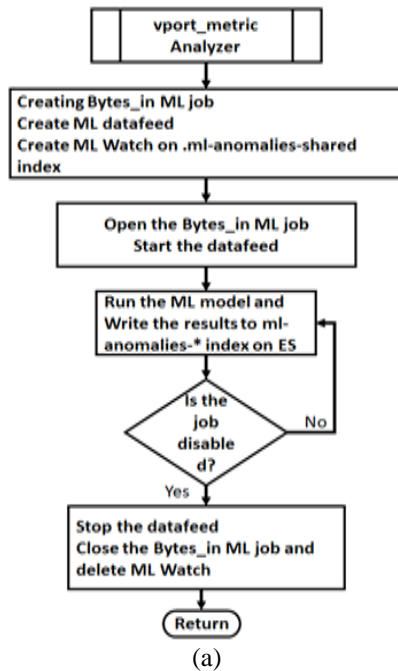


Figure 5. Creating vportId ML analyzer through VSD UI (a) flowchart for creation and (b) corresponding VSD UI

After creating the job, the *Bytes_in* ML job is opened and the datafeed is started, which provides the data to the detector by querying Elasticsearch. The ML job analyzes the data in real time obtained from the datafeed and continues to store results of the analysis by default in the `.ml-anomalies-shared` index with an anomaly score. The anomaly scores are given to the record based on the variation observed. The watcher gets triggered periodically and queries the `.ml-anomalies-shared` index to look for anomaly score crossing the configured threshold in the ES Watch. If one or more anomalies are found crossing the threshold, action is triggered on the ARE, which then processes the received data and takes appropriate action as defined as part of the ES watch action. The model keeps running until it is explicitly disabled by the user through VSD UI. Once disabled, the created job, datafeed and the watch are closed and deactivated.

The watcher gives an option to execute an action when anomalous event happens. A simple action could be a webhook for sending an alert message to the VSD UI. Other actions include moving the `vportId` to quarantine group to take further actions or blocking incoming flow to the `vportId`. It completely depends on the operators what they want to do. A sample JSON file listing is as shown in the Figure 6 for the multimetric job, where one job can be created with multiple detectors to look for anomalies in multiple metrics at the same time. The input records for these detectors are fetched from the same index of Elasticsearch specified in the datafeeds defined for the job.

```

1  "analysis_config":{
2  "bucket_span": "60s",
3  "detectors":[
4  {
5  "detector_description": "
6  BYTES_IN_detector",
7  "function": "mean",
8  "field_name": "bytes_in",
9  "partition_field_name": "vportId",
10 "detector_index": 0
11 },
12 {
13 "detector_description": "
14 BYTES_OUT_detector",
15 "function": "mean",
16 "field_name": "bytes_out",
17 "partition_field_name": "vportId",
18 "detector_index": 1
19 },
20 {
21 "detector_description": "
22 PACKETS_IN_detector",
23 "function": "mean",
24 "field_name": "packets_in",
25 "partition_field_name": "vportId",
26 "detector_index": 2
27 },
28 {
29 "detector_description": "
30 PACKETS_OUT_detector",
31 "function": "mean",
32 "field_name": "packets_out",
33 "partition_field_name": "vportId",
34 "detector_index": 3
35 }
36 ]
37 }

```

Figure 6. Sample JSON file listing for the `analysis_config`

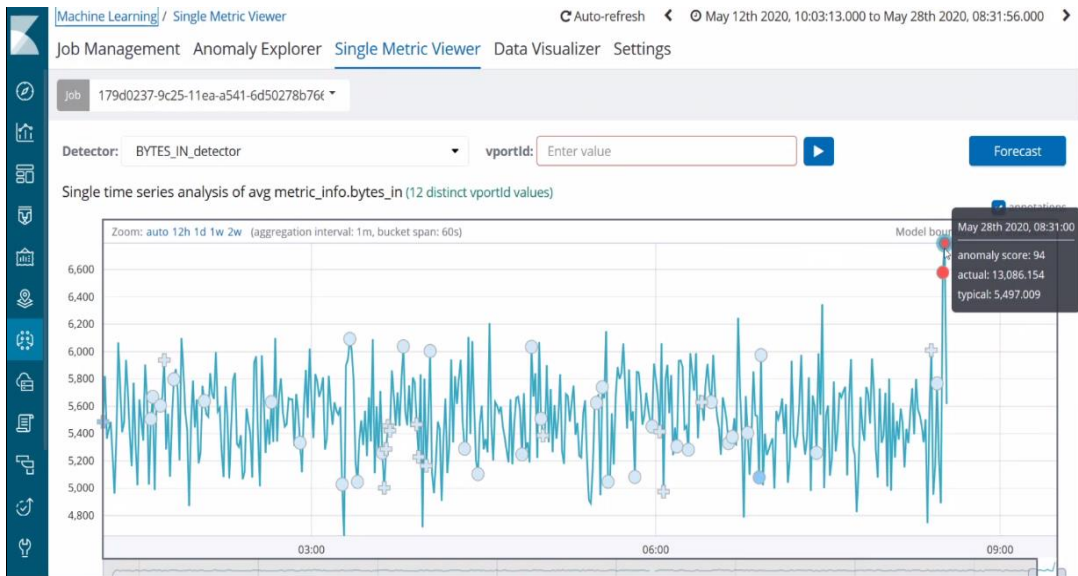
4. RESULTS AND DISCUSSION

Out of many experiments conducted, for the discussion of results two ML jobs are considered; one for the DoS attack created on BytesIn metrics for a specific `vportId` and one for the TCP flags to detect flood or DoS attacks. For creating the flood attacks, any of the 6 TCP flags could be used. For our experiment, we chose to focus on SYN flag. By focusing on one flag, we were able to streamline our approach and obtain focused results.

4.1. Case 1: vport metric ML analyzer job

Figure 7 shows the results of a simulation that involved 12 `vportIds` to connect source and destination nodes. In vport metric ML job, the anomalous behaviour was monitored on every `vportId`. Since the visualization is not yet built on to VSD, Kibana [27] is used to visualize the analysis done by the ML analyzer created. The normal flow was created with the packet size of 1,000 to 10,000 bytes. With this normal flow, a malicious flow was induced by sending packets with size 19,000 to 20,000 bytes to one of the `vportIds`,

4eef6134-7e47-11ea-ae91- bd028557e099, with randomly chosen source and destination IP addresses. Which suddenly increased the average BytesIn, giving a spike as shown in Figure 7(a), whose anomaly score is in the range for critical and the watch is accordingly configured to alert the user and take the actions. The corresponding record is as shown in the Figure 7(b).



(a)

time	max severity ↓	detector	found for	influenced by	actual	typical
May 28th 2020, 08:00	94	BYTES_IN_detector	4eef6134-7e47-11ea-ae91-bd028557e099	vportId: 4eef6134-7e47-11ea-ae91-bd028557e099	13,144.923	5,496.928
Description critical anomaly in BYTES_IN_detector found for vportId 4eef6134-7e47-11ea-ae91-bd028557e099						
Details on highest severity anomaly						
vportId	4eef6134-7e47-11ea-ae91-bd028557e099					
time	May 28th 2020, 08:30:00 to May 28th 2020, 08:31:00					
function	mean					
fieldName	metric_info.bytes_in					
actual	13145					
typical	5496.9					
job ID	179d0237-9c25-11ea-a541-6d50278b7661					
probability	1.9769959190089093e-24					
Influencers						
vportId	4eef6134-7e47-11ea-ae91-bd028557e099					

(b)

Figure 7. Vport metrics job in Kibana (a) spike showing the anomalous flow and (b) the corresponding record

The vport metric ML analyzer created was able to successfully detect the anomaly within 2min of creating the anomalous flows. Network Macro Group actions are defined for such cases where in the corresponding vportIds are moved to quarantine PG group for further monitoring. The VPort Metrics ML analyzer created was able to successfully detect the anomaly within 2min of creating the anomalous flows. Network Macro Group actions are defined for such cases where in the corresponding vportIds are moved to quarantine PG group for further monitoring.

The corresponding VSD UI is as in Figure 8. Time taken before an anomaly is detected involves writing the flow to the vport metric index, querying the index, and sending the data to ML analyzer, ML model analysis, writing the analysis with the anomaly score to .ml-anomalies-shared index, visualizing through Kibana. Since the VSD will not show the visualization and will take actions directly on the vportId, the time will be less than 2min compared to Kibana visualization.

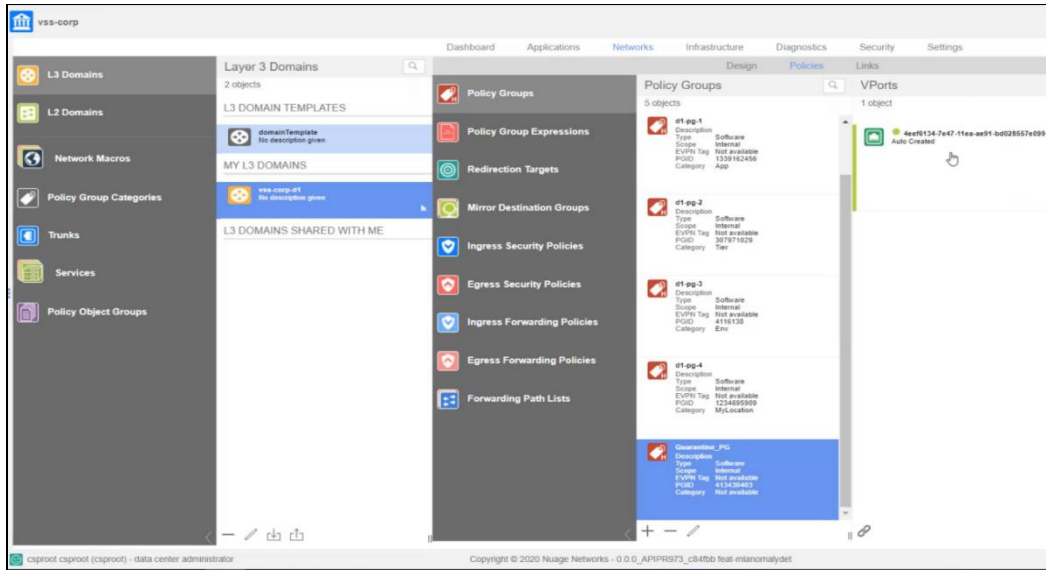


Figure 8. VSD UI highlighting the action quarantine PG taken on the anomalous vportId

4.2. Case 2: TCP flags ML analyzer job

The second type of ML analyzer was created for detecting flood attacks on TCP flags. TCP flags like SYN, RST, and FIN. can be good targets to make a flood attack to a particular destination. For testing a flood attack, TCP SYN flag is considered. A normal traffic is simulated between the randomly chosen source and destination IPs. Multiple detectors are created for detecting anomalies in TCP flags are combined in one TCP Flags ML analyzer. In normal pattern every destination would get a count of these flags between 1 to 2 at every destination. With this normal traffic, a TCP SYN flood attack was simulated between the source IP 10.2.0.143 and destination IP 20.2.0.210 by sending SYN requests between 5 to 6. This increased the count of SYN requests the destination was getting normally.

This increased the count of SYN requests the destination was getting normally. It was detected by the TCP Flags ML analyzer as shown in the Figure 9. The corresponding record is shown in Figure 10. The anomaly was found in less than 2min from the time the anomaly was introduced in the system. Network Macro actions are defined for such cases where in the corresponding sourceips are moved to quarantine IP group for further monitoring as shown in Figure 11. The TCP Flags ML analyzer can also detect the DDoS attacks on the destination IP using TCP flags, but the source IPs may not be found as it will be distributed. Since different attacks uses different patterns, some might suddenly increase the average traffic observed, some might suddenly decrease the traffic, in both cases ML Jobs thus created were successful in detecting the attacks.

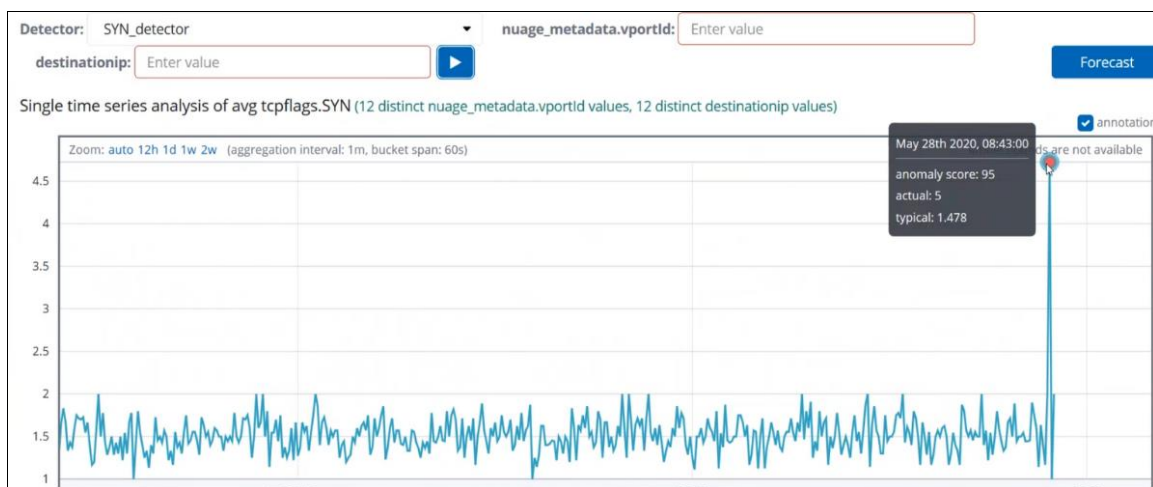


Figure 9. TCP flags ML analyzer anomalous pattern

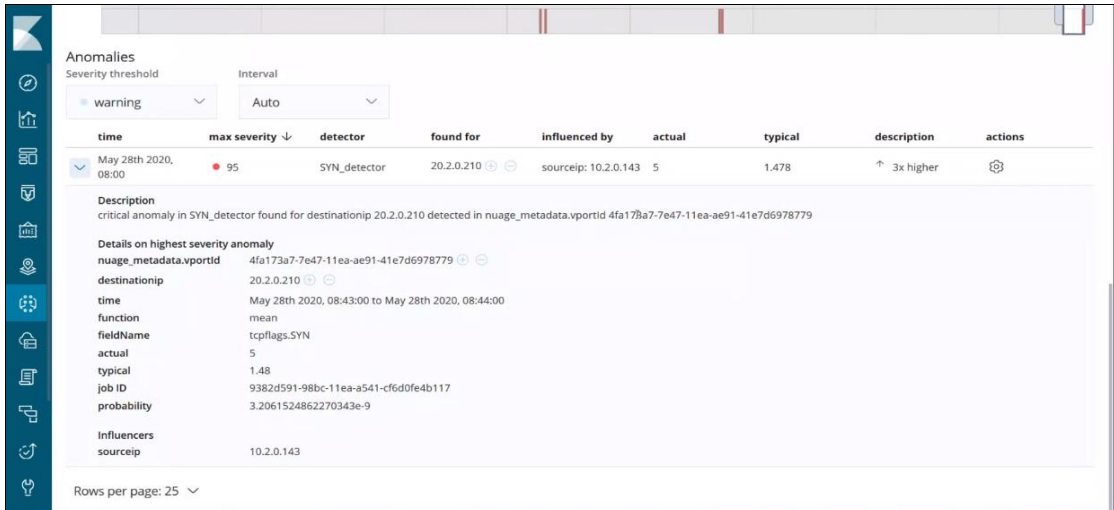


Figure 10. Description in Kibana for the TCP flags ML analyzer anomalous pattern

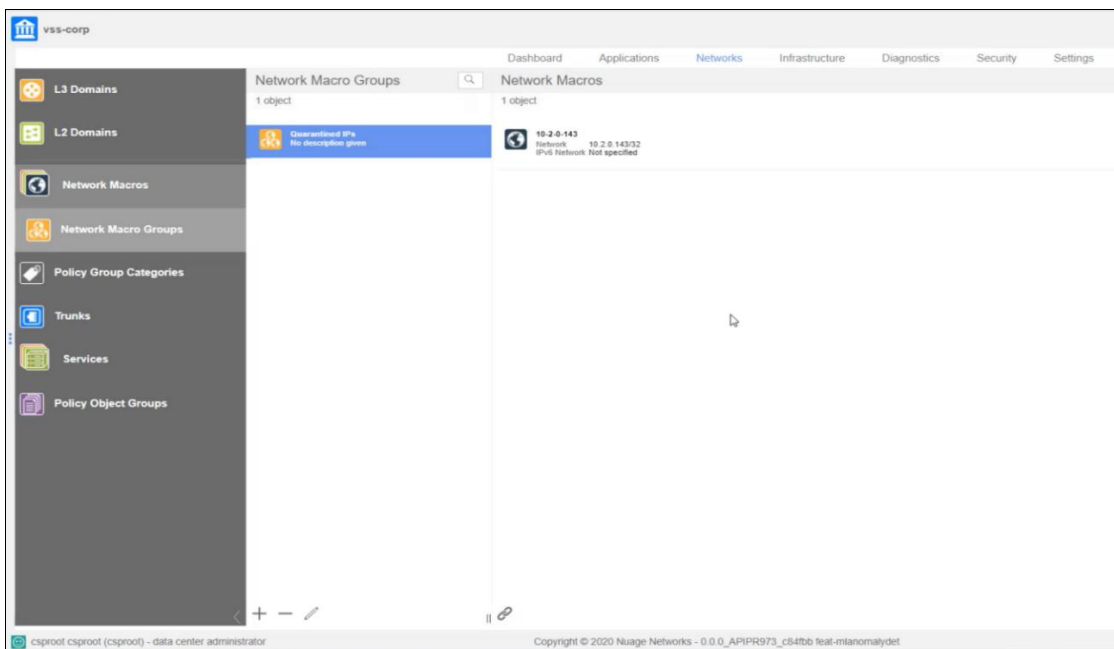


Figure 11. VSD UI showing action taken by moving the corresponding sourceip to Quarentine_IP

4.3. Limitations of the proposed system

The performance of the proposed system for anomaly detection is increased compared to the existing system which uses ARE for configuring a TCA to generate the alarms. There are few limitations or drawbacks of the proposed system:

- a) Proper values for the fields while creating the detectors/datafeed/ML analyzers are very important and users who do not have knowledge on these should be trained before they could use this system.
- b) Once the ML analyzer is created and the job is enabled until sufficient records are read to understand the normal pattern by the job, every record will be raised as an anomaly.
- c) Since the model is configured only through the APIs available, we barely know the algorithm for the model and will not be able to customize as per individual needs.
- d) Apart from all the above mentioned, the main limitation would be Elasticsearch X-Pack is proprietary and is not free.

5. CONCLUSION

The proposed system was developed using an SDN architecture integrated with Elasticsearch Machine Learning X-Pack libraries. A proof of concept was developed in Nuage Network's laboratory with one VRS node which simulates normal traffic using 12 vportIds and 10 IP addresses for source and destinations. And one VRS Node to simulate the anomalous traffic. The flow and vport statistics were recorded in the Elasticsearch indices. Compared to the existing model which sets static thresholds for observing anomalies, the proposed model used ML APIs to set the threshold dynamically based on the flow pattern observed. The results showed the anomalies being detected in near real time; the existing system was not able to detect. As a future work the authors are working on developing an anomaly detection ML model which will be able to detect the anomalies like Elasticsearch but will be opensource and can be used by anyone freely.

ACKNOWLEDGEMENTS

This work was carried out during the sabbatical training done by Prof.Sneha M in NOKIA Systems Pvt. Ltd. Bengaluru as part of her research, for the year 2019 to 2020. The authors would like to thank NOKIA Systems Bengaluru, India and the management of RV College of Engineering for providing this opportunity.




REFERENCES

- [1] Y. Li and Q. Liu, "A comprehensive review study of cyber-attacks and cyber security; Emerging trends and recent developments," *Energy Reports*, vol. 7, pp. 8176–8186, Nov. 2021, doi: 10.1016/J.EGYR.2021.08.126.
- [2] R. A. I. Alhayali, M. Aljanabi, A. H. Ali, M. A. Mohammed, and T. Sutikno, "Optimized machine learning algorithm for intrusion detection," *Indonesian Journal of Electrical Engineering and Computer Science (IJECS)*, vol. 24, no. 1, pp. 590–599, 2021, doi: 10.11591/ijeecs.v24.i1.pp590-599.
- [3] M. Suresh and R. Anitha, "Evaluating machine learning algorithms for detecting DDoS attacks," *Communications in Computer and Information Science*, vol. 196 CCIS, pp. 441–452, 2011, doi: 10.1007/978-3-642-22540-6_42/COVER.
- [4] P. S. Saini, S. Behal, and S. Bhatia, "Detection of DDoS attacks using machine learning algorithms," *Proceedings of the 7th International Conference on Computing for Sustainable Global Development, INDIACOM 2020*, pp. 16–21, Mar. 2020, doi: 10.23919/INDIACOM49435.2020.9083716.
- [5] K. Shaukat, S. Luo, V. Varadharajan, I. A. Hameed, and M. Xu, "A survey on machine learning techniques for cyber security in the last decade," *IEEE Access*, vol. 8, pp. 222310–222354, 2020, doi: 10.1109/ACCESS.2020.3041951.
- [6] N. P. Shetty, J. Shetty, R. Narula, and K. Tandona, "Comparison study of machine learning classifiers to detect anomalies," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 5, pp. 5445–5452, Oct. 2020, doi: 10.11591/IJECE.V10I5.PP5445-5452.
- [7] A. M. El-Shamy, N. A. El-Fishawy, G. Attiya, and M. A. A. Mohamed, "Anomaly detection and bottleneck identification of the distributed application in cloud data center using software-defined networking," *Egyptian Informatics Journal*, vol. 22, no. 4, pp. 417–432, Dec. 2021, doi: 10.1016/J.EIJ.2021.01.001.
- [8] T. Jafarian, M. Masdari, A. Ghaffari, and K. Majidzadeh, "A survey and classification of the security anomaly detection mechanisms in software defined networks," *Cluster Computer*, vol. 24, no. 2, pp. 1235–1253, Jun. 2021, doi: 10.1007/S10586-020-03184-1/TABLES/4.
- [9] A. M. Abdelrahman *et al.*, "Software-defined networking security for private data center networks and clouds: Vulnerabilities, attacks, countermeasures, and solutions," *International Journal of Communication Systems*, vol. 34, no. 4, p. e4706, Mar. 2021, doi: 10.1002/DAC.4706.
- [10] K. Farhana, M. Rahman, and M. T. Ahmed, "An intrusion detection system for packet and flow based networks using deep neural network approach," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 5, pp. 5514–5525, Oct. 2020, doi: 10.11591/IJECE.V10I5.PP5514-5525.
- [11] M. Al-Shabi and A. Abuhamdah, "Using deep learning to detecting abnormal behavior in internet of things," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 12, no. 2, pp. 2108–2120, Apr. 2022, doi: 10.11591/IJECE.V12I2.PP2108-2120.
- [12] O. Negoita and M. Carabas, "Enhanced security using Elasticsearch and machine learning," in *Advances in Intelligent Systems and Computing*, 2020, doi: 10.1007/978-3-030-52243-8_19.
- [13] G. Folino, C. O. Godano, and F. S. Pisani, "An ensemble-based framework for user behaviour anomaly detection and classification for cybersecurity," *Journal of Supercomputing*, pp. 1–24, Jan. 2023, doi: 10.1007/S11227-023-05049-X/TABLES/8.
- [14] A. Hariharan, A. Gupta, and T. Pal, "CAMLPAD: cybersecurity autonomous machine learning platform for anomaly detection," in *Advances in Intelligent Systems and Computing*, 2020, doi: 10.1007/978-3-030-39442-4_52.
- [15] "Home - Nuage Networks." <https://www.nuagenetworks.net/> (accessed Mar. 15, 2023).
- [16] B. I. Farhan and A. D. Jasim, "Performance analysis of intrusion detection for deep learning model based on CSE-CIC-IDS2018 dataset," *Article in Indonesian Journal of Electrical Engineering and Computer Science*, vol. 26, no. 2, pp. 1165–1172, 2022, doi: 10.11591/ijeecs.v26.i2.pp1165-1172.
- [17] H. Kamel and Z. Abdullah, "A new approach of extremely randomized trees for attacks detection in software defined network," *Indonesian Journal of Electrical Engineering and Computer Science (IJECS)*, vol. 28, no. 3, pp. 1613–1620, 2022, doi: 10.11591/ijeecs.v28.i3.pp1613-1620.
- [18] M. I. Kareem, M. Jasim, and N. Jasim, "Entropy-based distributed denial of service attack detection in software-defined networking," *Indonesian Journal of Electrical Engineering and Computer Science (IJECS)*, vol. 27, no. 3, pp. 1542–1549, 2022, doi: 10.11591/ijeecs.v27.i3.pp1542-1549.
- [19] "Elasticsearch Guide [8.6] | Elastic." <https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html> (accessed Mar. 15, 2023).
- [20] M. Komisarek, M. Pawlicki, R. Kozik, and M. C. Chora's, "Machine learning based approach to anomaly and cyberattack detection in streamed network traffic data," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 12, no. 1, pp. 3-19, 2021, doi: 10.22667/JOWUA.2021.03.31.003.




- [21] N. Shah, D. Willick, and V. Mago, "A framework for social media data analytics using Elasticsearch and Kibana," *Wireless Networks*, vol. 28, no. 3, pp. 1179–1187, Apr. 2022, doi: 10.1007/S11276-018-01896-2/FIGURES/4.
- [22] C. Gormley and Z. Tong, "Elasticsearch the definitive guide: a distributed real-time search and analytics engine," *O'Reilly Media, Inc.*, 2015.
- [23] U. Thacker, M. Pandey, and S. S. Rautaray, "Performance of elasticsearch in cloud environment with nGram and non-nGram indexing," *International Conference on Electrical, Electronics, and Optimization Techniques, ICEEOT 2016*, 2016, pp. 3624–3628, doi: 10.1109/ICEEOT.2016.7755381.
- [24] M. S. Divya and S. K. Goyal, "ElasticSearch an advanced and quick search technique to handle voluminous data," *An International Journal of Advanced Computer Technology*, vol. 2, no. 6, 2013.
- [25] "Create anomaly detection jobs API | Elasticsearch Guide [8.6] | Elastic." <https://www.elastic.co/guide/en/elasticsearch/reference/current/ml-put-job.html> (accessed Mar. 17, 2023).
- [26] "How Watcher works | Elasticsearch Guide [8.6] | Elastic." <https://www.elastic.co/guide/en/elasticsearch/reference/current/how-watcher-works.html> (accessed Mar. 17, 2023).
- [27] "Kibana Guide [8.4] | Elastic." <https://www.elastic.co/guide/en/kibana/8.4/index.html> (accessed Mar. 17, 2023).

BIOGRAPHIES OF AUTHORS






Sneha Mahabaleshwar    received the B.E degree in Computer Science and Engineering from R N Shetty Institute of Technology, Bangalore, India, in 2006 and M Tech degree in Computer Networking Engineering from S J B Institute of Technology, Bangalore, India, in 2013. Currently, she is working as Assistant professor in the Department of Computer Science and Engineering, R. V. College of Engineering, Bengaluru, India. Her research interests include computer networks, software defined networks, datacenter networks, and application of machine learning algorithms for computer network problems. She has completed sabbatical training at NOKIA Systems, India, as part of her research in the year of 2019 to 2020. She can be contacted at email: sneham@rvce.edu.in.



Shobha Gangadhar    she is a Professor, Computer Science and Engineering Department, R. V. College of Engineering, Bengaluru, India have teaching experience of 25 years, her specialization includes data mining, machine learning and image processing. She has published more than 150 papers in reputed journals/conferences. She has also executed sponsored projects worth INR 200 lakhs funded from various agencies nationally and internationally. She is a recipient of various awards such as Career Award for young teachers 2007-08 constituted by All India Council of Technical Education, Best Researcher award from Cognizant 2017, GHC Faculty Scholar for Women in Computing in 2018, IBM Shared University Research Award in 2019, HPCC Systems community recognition award 2020. She can be contacted at email: shobhag@rvce.edu.in.



Sharath Krishnamurthy    he was graduated with a B.E. degree in Electronics and Communication from Malnad College of Engineering, Mysore University, India, in 1999 and M.S in Software Systems from Birla Institute of Technology and Science, Rajasthan, India in 2003. He started his career as a Software Engineer at Robert Bosch India Ltd. working on network management of Bosch PBXs for Bosch Telecom. He has a vast experience in the field of network management which include multi-service provisioning devices, CDMA RAN, IMS networks, IP/Optical networks, DWDM optical transport, IP routing and services for access/edge and core IP routers spanning across organizations like metro-optix, lucent technologies, alcatel-lucent and nuage networks where he was an architect for the analytics platform which included traffic flow collection and telemetry stats for nuage networks VCS and VNS solutions. Currently he is working as a principal technical specialist at Nokia India working on data center management solutions on a new product line. He can be contacted at email: sharath.k@nokia.com.