2021 International Symposium on Networks, Computers and Communications (ISNCC) | 978-1-6654-0304-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISNCC52172.2021.9615638

API Security in Large Enterprises: Leveraging Machine Learning for Anomaly Detection

Gaspard Baye*, Fatima Hussain†, Alma Oracevic‡, Rasheed Hussain*, and S.M. Ahsan Kazmi*

* Networks and Blockchain Lab, Innopolis University, Innopolis 420500, Russia
Email: {b.gaspard, r.hussain,a.kazmi}@innopolis.ru
† Royal Bank of Canada, Toronto, Canada
Email: fatima.hussain@rbc.com
‡ University of Bristol, Bristol, UK

***Abstract*—Large enterprises offer thousands of micro-services applications to support their daily business activities by using Application Programming Interfaces (APIs). These applications generate huge amounts of traffic via millions of API calls every day, which is difficult to analyze for detecting any potential abnormal behaviour and application outage. This phenomenon makes Machine Learning (ML) a natural choice to leverage and analyze the API traffic and obtain intelligent predictions. This paper proposes an ML-based technique to detect and classify API traffic based on specific features like bandwidth and number of requests per token. We employ a Support Vector Machine (SVM) as a binary classifier to classify the abnormal API traffic using its linear kernel. Due to the scarcity of the API dataset, we created a synthetic dataset inspired by the real-world API dataset. Then we used the Gaussian distribution outlier detection technique to create a training labeled dataset simulating real-world API logs data which we used to train the SVM classifier. Furthermore, to find a trade-off between accuracy and false positives, we aim at finding the optimal value of the error term (C) of the classifier. The proposed anomaly detection method can be used in a plug and play manner, and fits into the existing micro-service architecture with little adjustments in order to provide accurate results in a fast and reliable way. Our results demonstrate that the proposed method achieves an F1-score of 0.964 in detecting anomalies in API traffic with a 7.3% of false positives rate.**

***Keywords*—Application Programming Interface (API), Machine Learning, Support Vector Machine (SVM) , Micro-service, Anomalies**

I. Introduction and Related Work

Data breaches cost large enterprises staggering amounts of money. The average cost of a data breach is $3.86 Million (according to IBM, 2020 [1]). APIs are essential to resource and utility sharing and enabling business expansion in large enterprises. According to reports from Akomai in 2019 [2], API calls represented 83% of all the internet traffic. Another study from Gartner [3] predicted that by 2021, exposed APIs without adequate security measures will lure cyber attackers to 90% of web-enabled applications. Due to many attacks observed on APIs, the Open Web Application Security Project (OWASP) created a special project dedicated for APIs (OWASP API Security Top 10) to report the most recurrent/popular attacks on APIs. These OWASP API logs serve as a reference document for API developers, system analysts, and any person interested in API security to understand the patterns of these attacks [4]. Therefore, in the face of such attacks on APIs, it is essential to consider API security and detect anomalies in APIs. In essence, APIs are easy targets for cyber-attackers to get an access to a system. Therefore, traffic classification mechanisms have been extensively used to cater the challenges of API vulnerabilities. According to Bashir et al. [5], Intrusion Detection System (IDS) and Intrusion Prevention System (IPS) are the preferred choices to limit unauthorised access to APIs provided by the large enterprises. Existing IDS and IPS solutions use rule-based and signature-based techniques to classify access to APIs. However, these methods do not account for the unseen attacks and malicious patterns. In other words, these techniques do not monitor malicious behaviors within the API traffic.

Given the dynamism and variety of new cyberattacks, it is imperative to develop intelligent solutions that take into account malicious behaviours and monitors unseen attack patterns within the API traffic. In this regard, Machine Learning (ML) techniques can be a suitable choice to address the API security challenges. To date, several studies have proposed different ML techniques detect anomalies in the network traffic. For example, Peter et al. [6] proposed an analytical framework to analyze and detect anomalies for network traffic data. In their study, they compared the rate of false positives and prediction accuracy for different ML models (like the ensemble model, the Seasonal Autoregressive Integrated Moving Average (SARIMA), times series model and Long Short-Term Memory (LSTM) Autoencoder model). Their ensemble model is comprised of SARIMA and LSTM, and the ensemble produced results with a true positive rate of 14.58% and a false positive rate of 1.7%. The ensemble model showed promising results in reducing false positives using a method which only classified data as anomalous data only if both models flag it as anomalous. With this approach, it succeeded to reduce the false positive rate of the auto-encoders by 50%. However, this research was conducted specifically for Capital One company using capital One specific data. As a result, this technique may work well only for capital One specific data but might fail to work for non Capital One specific data.

In another study, Yan et al. [7] proposed a model to solve the anomaly problem in the general network traffic via the use of entropy theory and Support Vector Machine (SVM). Their method uses entropy theory to measure the degree of randomness in the network traffic and define their features. These features are later used by the SVM classifier to classify the traffic as anomalous or genuine. They trained on KDD-CUP 99 data, a popular dataset for network traffic based on specific traffic features such as source IP, destination IP, source and destination port, packet size and type. Their results showed that the proposed technique is efficient as compared to other existing methods. However, their technique was only used for general network traffic which usually constitutes many different sub traffics like DNS, ARP, TELNET, VLAN tags and many more, all of which have diverse behaviors in a network. While API traffic is specific with unique features and behaviors that can differ from other network traffic, this technique may work well on general network traffic but not for API traffic. Several other works such as [8]–[13] have studied network traffic anomaly detection and have reported failures when applied to API traffic due to the unique nature and behavior of API traffic.

As mentioned above, anomaly detection in API traffic calls for intelligent mechanisms and ML is leveraged to obtain convincing results in terms of detection accuracy and low false positive rates. The main goal of using ML techniques is to develop a model that can learn from its experience and make predictions without being explicitly programmed [14]. More precisely, ML models, especially supervised learning gain their experience from labelled datasets. The algorithm learns normal and abnormal behaviour from its predefined datasets.

To date, there have been very few studies that address API security. For example, Fatima et al. [15] proposed a mechanism for secure API service mesh with Istio and Kubernetes. They added a smart way to associate new API to an already existing category of service mesh. Their techniques do not detect anomalies within the API traffic using ML model but their approach proposes a secure framework and implementation for APIs. Another method was proposed by Jyothi et al. [16] to implement restful APIs in a secure micro-service architecture, as well as secure back-end practices.

A. Our Contributions

Our contribution in this area is to use the Gaussian distribution approach for outlier detection. This approach uses the standard deviation as a way to statistically measure the dispersion of data from the mean. We use this technique for our API traffic to determine outliers, i.e., traffic very far away from the mean. In order to achieve this, we first create labelled data by selecting data within the range of a pre-calculated standard deviation which we will define as normal traffic. The outliers that fall far away from the mean, we define them as abnormal traffic. We then analyze the API traffic using cognitive techniques by means of SVM to thoroughly analyze and detect abnormal patterns in API traffic, classifying them for the analyst to further investigate. We apply the confusion matrix to reduce the false positive alerts to report only true positive events. Because this algorithm can analyze and visualize extremely large traffic, it reduces the cognitive load on the security analyst. We would like to make a general remark that this work does not aim to create a tool which can be directly used in micro-service architectures. Instead, we focus on creating a Proof of Concept (PoC) that demonstrates the ability to accept data from different text sources like logs, analyze it using SVM algorithm and classify this traffic as being normal or abnormal. The research questions we want to answer, are as follows:

- *Can ML-based techniques be used to efficiently analyze and detect anomalies in API traffic?*

To answer this question, we broke it into two smaller questions as follows.

1) *Can we trust the accuracy and analysis made by the employed ML algorithm?*
2) *How fast is the analysis process? Can this be done in real time?*

SVM is a very effective but yet simple algorithm. Achieving local optima in SVM is usually after few iterations as compared to other deep learning algorithms. SVM also works well for classification problems with datasets having few parameters. Based on the structure of API traffic with few parameters, we selected SVM which stood out to be the best for API analysis as well as from many other surveys [7], [8], [17]–[21] which used this algorithm with similar data. SVM can be implemented using the python library called Scikit-learn. Scikit-learn published a chart that could help us chose the right estimator for our case. The chart can be seen in Fig. 1. Since we need a predicting category, we have less than 100k sample data which we labeled, and can confidently use SVM with a linear kernel (SVC) as our model. The rest of this paper is organized as follows: In Section II, we discuss the API dataset and its key features. In Section III, we show how we model the API dataset for SVM classification. In Section IV, we discuss our findings, how the model performs, the accuracy, the time required to classify API traffic and compare it with rule-based techniques. Finally, Section V concludes the paper with a discussion on future works.

II. API Traffic Analysis and Key Performance Indicators

Large enterprise organizations use micro-service architecture. This architecture mostly uses Restful APIs to provide communication between clients, API gateway (reverse proxy, security, load balancing), services like authentication, and data stores usually called database. In this section, we discuss the API dataset. We elaborate on the technique we used to create the training dataset and the different features of this dataset. We also visualize the state of the dataset to determine whether it is a balanced or unbalanced dataset. This will help us determine state-specific metrics that will be used for testing and validation.

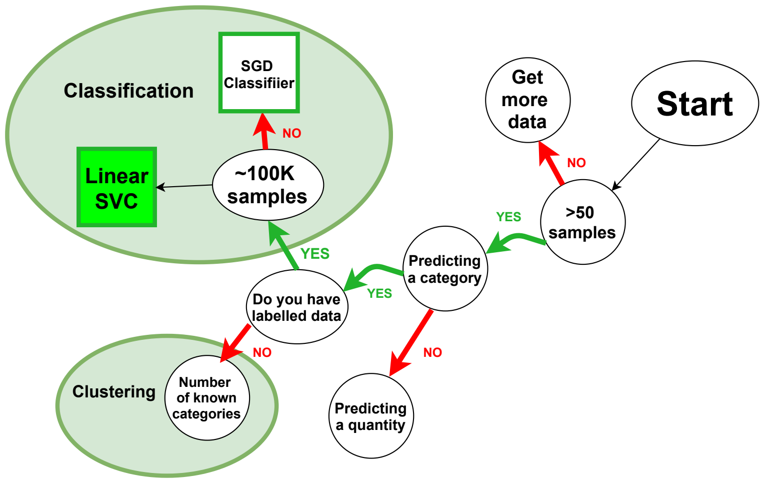

Fig. 1. Choosing the right estimator

A. Data Generation

We generate our own dataset by using synthetic techniques inspired from real-world API traffic. This is because labelled dataset of API traffic is not readily available due to intellectual property or privacy related issues.

Initially, we considered publicly available datasets such as the dataset from the Knowledge Discovery and Data Mining Tools Cup (KDD Cup) from 1999; however, this dataset does not satisfy the specifications of API traffic data. This data is meant for general network traffic and not specific for API traffic. Furthermore, this dataset is outdated and does not fit in our context. Therefore, we developed our dataset by using synthetic techniques inspired by real-world API traffic where we defined normal data as the traffic within the standard deviation range (σ) of a Gaussian or normal distribution of the entire traffic. Data points falling out of this range are called outliers and are labeled as abnormal traffic. The resulting dataset has well-defined features that we will discuss in detail in the feature engineering section.

B. Feature Engineering and KPIs

API traffic, just like other network traffic has characteristics that uniquely describe them. We need to exploit these features to use them to train ML models. In this paper, we consider 6 features related to API traffic. These features include the client IP address, the HTTP request type, the HTTP code, the number of consecutive requests, the bandwidth of the request, and the connection-token. In a Distributed Denial of Service (DDoS) attack for example, we can observe amplified versions of features like the numbers of client IP addresses, the bandwidth consumption, the numbers of consecutive requests, connection duration, and numbers of HTTP requests. In the following, we elaborate more about these features and how these features can be used for the classification.

1) Bandwidth consumption: Bandwidth is the maximum transfer rate in a communication channel. A microservice serves data to a client IP at a particular rate. There is always a specific allowable rate of data transfer set by the system administrator. Any request that exceeds this allowable value ϵ is considered abnormal. In this paper, we consider these parameters in our ML model. It is worth mentioning that by bandwidth we mean the communication between two parties particularly using IP backbone.

2) Client IP address: An IP address is a numerical label given to every computer that communicates in a network. It is used to identify one device from the other in a network. So, any device that requests specific data will have its IP address logged with the data the device requested. IP address, at least in part, helps in keeping track of the device activity on the network. For instance, in case of a DoS attack, the attackers usually trigger requests from many devices in the network and these devices communicate with their IP addresses. Due to the large volume of requests, huge bandwidth is generated, usually more than what the service can support. This results in service crashing causing an outage. These different devices' IP addresses help us to know the total number of consecutive requests performed by a specific IP address per specific time frame.

3) Number of consecutive requests: The number of requests per unit time performed by an address during a peak transaction period is not supposed to pass a certain constraint ϵ defined per that period (as per policy). If we observe a sudden change in that state, there might be an abnormal activity and security analysts need to take a closer look into that. In this work, we will pay close attention to, and analyze this feature in our ML algorithm.

4) Connection duration (token size): This is usually the maximum acceptable time for a client to access a resource or consume an API. We consider this because in certain attack scenarios, for example, in a DoS attack scenario, the attacker tries as much as possible, to consume more bandwidth and resources, thus trying to consume the API for a much longer period so that legitimate requests would not be successful. And this is usually done through different HTTP requests.

5) HTTP requests: In API standards, there exist many HTTP requests. Some of them include GET, POST, DELETE, PUT, OPTIONS, HEAD, PATCH, and TRACE. The majority of these requests are attached to response codes such as 200 for a successful transaction and 404 for file not found. Both HTTP methods and HTTP response codes are used in Restful APIs to determine the status of the request. We consider a scenario where a device is used to perform a GET request at a certain time of the day suddenly change and perform a POST or a DELETE request. This can be a possible abnormal situation that should be addressed.

Once our dataset is defined, we analyze our dataset using dataset evaluation metrics like data visualization. This technique will graphically present the state (balanced or imbalanced) of our dataset. Data visualization techniques reveal that our dataset is imbalanced. This means accuracy is not the best measure of performance of our model. We will employ techniques to convert this imbalance dataset into a balanced dataset by using the right evaluation metrics like confusion matrix, precision, Recall, and F1-score which will be detailed in Section III-D.

III. Classification and Validation

In this section, we show how we model the API dataset for SVM classification. First we discuss the Gaussian distribution followed by the SVM classifier and training. Then we discuss in detail, the testing and validation of our techniques for an unbalanced dataset.

A. Gaussian Distribution

We use Gaussian distribution approach to create the training dataset which we use to train the SVM classifier. In our approach, we assume that each data point in the dataset will follow the probability distribution with a graphical representation of a bell-shape. This graphical representation is based on the variance (σ^2), mean (μ) and standard deviation (σ) of our dataset. These terms help us to determine the probability distribution $P(x)$ of the dataset. The standard deviation uses the mean to measure the dispersion of the dataset by determining the relative distance of this dataset to the mean. Moreover, the standard deviation is calculated from the square of the variance. We use the standard deviation (σ) to know the region where majority of data points are located. From that, we can quickly find out data points that fall far apart from the standard deviation. We label these points as outliers or anomalies.

Let us assume there are m $\{1, 2, ..., M\}$ instances of the API traffic data points in our dataset, each with j selected features (number of successive requests per token N and bandwidth B).

The mean (μ) for every feature $\{j$=1, 2,…, $N\}$ at position $\{i$=1, 2, …, $M\}$ is given by Eq. 1 below:

$$\mu = \frac{1}{M}\sum_{i=1}^{M} x_j^i \tag{1}$$

where i is the position of the x_j data point within the m training set. The variance σ_j^2 is given by the Eq. 2 below.

$$\sigma^2 = \frac{1}{M}\sum_{i=1}^{M}(x_j^i - \mu)^2 \tag{2}$$

The probability $P(X)$ for a random dataset X is given by Eq. 3.

$$P(X) = \prod_{j=1}^{N} P(X;\mu,\sigma^2) \tag{3}$$

We can select a certain constraint, ϵ, defined by system admins or security analysts. The constraint value will determine whether the data is anomalous or genuine by following the constraint given by Eq. 4:

$$\begin{cases} Anomaly & \sigma < \epsilon < \sigma + k\sigma, \forall k \in \mathbb{R}^+ \\ Normal & Otherwise \end{cases} \tag{4}$$

B. Training

The Gaussian distribution approach based on the constraint defined in (eq. 4), enables us to create and label our training dataset. This dataset can be much more complex in a real-life situation where system admins may have many other features to describe their datasets. For this paper, we used the simplest case to demonstrate our technique.

C. Used Classifier: Support Vector Machine (SVM)

SVM is a supervised ML algorithm that optimally classifies data points using a hyper-plane in a high-dimensional space. SVM can be used for classification or outlier detection. This categorization depends on the number of classes to separate. A line separates two classes, i.e., a binary classification has a straight line as a class separator. Good separation is achieved when the hyper-plane is at the largest distance with the support vectors that constitute training vector points at the functional margin. More often, the best accuracy is observed from classifiers with hyper-planes having the largest margins.

SVM uses parameters like the regularization, kernel, and the margin to reduce its computational load by adjusting them to improve the model performance. SVM supports many modes depending on the complexity of the dataset that the kernel has to separate. Different modes of SVM classification include linear kernel, polynomial, and Radial Basis Function (RBF) modes. In this paper, we implement SVM with a linear kernel. In our case, the training data was grouped into two class types, i.e., anomalous class representing traffic with anomaly denoted as C^-, and the normal class representing normal traffic denoted as C^+. Suppose we have a training dataset x, having m samples and D features such that x_j^i is the i^{th} sample, j^{th} feature. The vector $\vec{x_i}$ describes the total D features for the data point at position i. The label vector $\vec{y_i}$ has two values, one of them C^+ representing the predicted normal traffic and the other one C^-, representing the predicted abnormal traffic. The hyper-plane is represented using the Eq. 5 below.

$$\vec{w}.\vec{x} + b = 0, \forall b \in \mathbb{R}^D \tag{5}$$

where the vector $\vec{w}$, $\vec{w}$ = $[\beta_0,\beta_1]$ is perpendicular to the hyper-plane and $\vec{x}$, $\vec{x}$ = $[x_1,x_2]$ lies on the hyper-plane. The width of the margin can be determined from the pair of support vectors found in the two classes, C^+ and C^- represented by $\vec{X^+}$ and $\vec{X^-}$ respectively, projected onto $\vec{\omega}$. The dot product of the unit vector in the direction of $\vec{\omega}$ and $(\vec{X^+} - \vec{X^-})$ gives us the equation of the margin, i.e., Eq. 6 as given below:

$$(\vec{X^+} - \vec{X^-}).\frac{\vec{\omega}}{||\vec{\omega}||} = \frac{2}{||\vec{\omega}||} \tag{6}$$

The main objective of SVM is to find the optimal hyper-plane to separate the normal class C^+ and the abnormal class C^- as shown in Fig. 2. This leads us to the following optimization problem which can be achieved via the following Lagrange function (L).

$$min\{\frac{1}{2}||\omega||^2 - \sum \alpha_i[y_i(\vec{\omega}.\vec{x_i} + b) - 1]\} \tag{7}$$

At maximum width, $\nabla L = 0$ which implies:
$\frac{\partial L}{\partial \varphi} = 0$, we will get $\vec{\omega} = \sum_i \alpha_i y_i x_i$
$\frac{\partial L}{\partial b} = 0$, we will get $\sum_i \alpha_i y_i = 0$
After assembling the different pieces, the complete Eq. (8) will result to :

$$L = \sum \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \vec{x_i}.\vec{x_j} \tag{8}$$

We can observe from Lagrange optimization that the terms that matter here are $(\vec{x_i}.\vec{x_j})$. This means that the SVM classifier depends only on this dot product of the sample. The parameter α directly affects the width of the margin. The larger the value of α, the wider is the functional margin.

In soft margin SVM, the error term C_i tells us about the optimization of how much we want to avoid misclassification of each training sample. This can be seen in Eq. 9 below.

$$y_i(\vec{\omega}.\vec{x_i} + b) = 1 - C_i \tag{9}$$

D. Testing and validation

We tested our proposed anomaly detection scheme using real-world data in collaboration with industry experts. We used the performance in the training set and compared it with that of the test set. The Recall, the ratio of the total number of correctly classified examples belonging to a class with respect to the total number of examples belonging to that class. High Recall value indicates that the class is correctly categorized, i.e., small number of False Negative (FN).

To get the value of the precision, we divide the total number of correctly classified positive examples by the total number of the predicted positive examples. High precision indicates an example labelled as positive is indeed positive (small number of FP).

From the Recall and Precision, we calculated the F-score. The F-score tell us in a more accurate way how the model is performing. An excellent model have an F-score of 1.

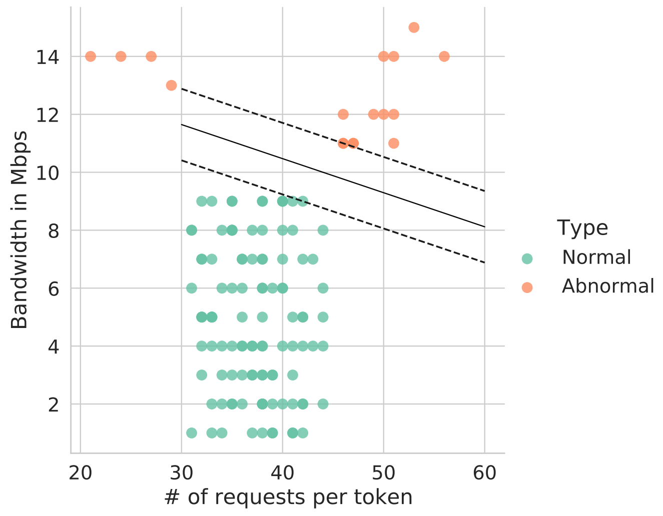

Fig. 2. Classified data set with SVM

IV. Experiments and Results

In this section, we discuss our findings based on the already discussed research questions.

From our synthetic data generated, we processed 98,000 requests for 255 IP addresses in one connection. These 98,000 requests were used to create a training dataset through the Gaussian approach based on the constraint in Eq. 4.

We simulated many instances of stealth DoS attack inspired by real-world attacks like the CVE-2011-3192 DoS attack on the Apache server. This attack could bypass the DoS rules set by popular IDS like Snort and Suricata. During this attack, we created a request with overlapping byte ranges. When Apache saw this request or Request-Range HTTP headers, each range was treated as a new request. Memory was confidently allocated for the different ranges so that it can be processed and responded to, accordingly. We sent hundreds of packets containing lots of ranges which we observed memory consumption of about 1GB/10 seconds. We evaluated the detection rates of Snort and SVM-based technique and compared the results as shown in Table I. We also applied SVM to the test dataset. In particular, we varied the error term C and observed the behavior of the model. Different values of C were used ranging from C = 0.1, to C = 50. The obtained results based on the above parameters, are shown in Fig. 3. It can be observed that the F1-score decreases with an increased value of C. On the other hand, we can also observe that the SVM training time, testing time, and classification time decreases as C increases. This is because increasing the C-parameter value increases over-fitting and makes the hyper-plane misclassify certain anomalous data points causing a lower F1-score as we increase the value of C. We can also observe that the SVM algorithm requires less amount of time for training, testing, and classification. Classification time is in orders of seconds after sufficiently training the model on a 16G RAM machine with NVIDIA GEFORCE GTX 960M. Utilizing it for a real-time process with a more sophisticated machine will require even

TABLE I
THIS TABLE SHOWS COMPARISON BETWEEN RULE BASED AND SVM BASED APPROACH

Method	Unseen detected(%)	FP(%)	TP(%)	F-score
SVM-based	**83.5**	**7.3**	**92.7**	**0.964**
Snort	20.45	23.5	76.5	0.751
Suricata	22.45	21.5	77.5	0.761

less time. SVM is a highly optimized algorithm that requires few seconds to classify around 100k dataset.

In Table I, results show the detection rates for both methods based on SVM and rule-based mechanisms for various evaluation metrics used for unseen attacks, False Positive (FP), False Negative, (FN), and F-score. It can be observed that SVM performs better on unseen attacks than rule-based techniques using Snort and Suricata. This is because CVE-2011-3192 is a new type of attack which the IDS did not see before and complex to detect from the rules. Our ML-based method has low false positives rates compared to snort, making it suitable for usage in the real world.

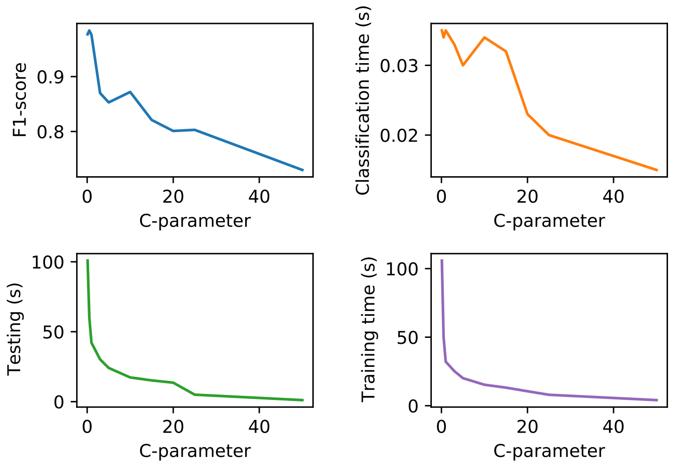

Fig. 3. Variation of different model entities versus C-terms

V. CONCLUSION

In this paper, we leveraged SVM-based technique for anomaly detection in API traffic. Our experimental results show that the proposed ML-based technique performs better by providing more precise results in terms of anomaly detection in API traffic than the existing rule-based methods. It also shows that ML-based technique is faster in classifying API traffic as abnormal or normal with low false positives. This makes it very suitable to be used in real-world settings and automate the API security in enterprises.

REFERENCES

[1] I. S. International Business Machines. (2020) 2020 cost of a data breach report. [Online]. Available: https://www.ibm.com/security/digital-assets/cost-data-breach-report/#/

[2] Akamai. (2019) State of the internet / security | retail attacks and api traffic (volume 5, issue 2) | akamai. [Online]. Available: https://www.akamai.com/uk/en/multimedia/documents/state-of-the-internet/state-of-the-internet-security-retail-attacks-and-api-traffic-report-2019.pdf

[3] Gartner. (2019) Gartner's api strategy maturity model. [Online]. Available: https://www.gartner.com/en/documents/3970520/gartner-s-api-strategy-maturity-model

[4] T. O. W. A. S. Project(OWASP). (2019) OWASP API Security Top 10the ten most critical api security risks. [Online]. Available: https://owasp.org/www-project-api-security/

[5] U. Bashir and M. Chachoo, "Intrusion detection and prevention system: Challenges & opportunities," in *2014 International Conference on Computing for Sustainable Global Development (INDIACom)*. IEEE, 2014, pp. 806–809.

[6] P. Kromkowski, S. Li, W. Zhao, B. Abraham, A. Osborne, and D. E. Brown, "Evaluating statistical models for network traffic anomaly detection," in *2019 Systems and Information Engineering Design Symposium (SIEDS)*. IEEE, 2019, pp. 1–6.

[7] G. Yan, "Network anomaly traffic detection method based on support vector machine," in *2016 International Conference on Smart City and Systems Engineering (ICSCSE)*. IEEE, 2016, pp. 3–6.

[8] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Computing*, pp. 1–13, 2019.

[9] B. J. Radford, L. M. Apolonio, A. J. Trias, and J. A. Simpson, "Network traffic anomaly detection using recurrent neural networks," *arXiv preprint arXiv:1803.10769*, 2018.

[10] T.-Y. Kim and S.-B. Cho, "Web traffic anomaly detection using c-lstm neural networks," *Expert Systems with Applications*, vol. 106, pp. 66–76, 2018.

[11] H. H. Pajouh, G. Dastghaibyfard, and S. Hashemi, "Two-tier network anomaly detection model: a machine learning approach," *Journal of Intelligent Information Systems*, vol. 48, no. 1, pp. 61–74, 2017.

[12] M. Zhu, K. Ye, Y. Wang, and C.-Z. Xu, "A deep learning approach for network anomaly detection based on amf-lstm," in *IFIP International Conference on Network and Parallel Computing*. Springer, 2018, pp. 137–141.

[13] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, 2016, pp. 21–26.

[14] T. Huizinga, "Using machine learning in network traffic analysis for penetration testing auditability," 2019.

[15] F. Hussain, W. Li, B. Noye, S. Sharieh, and A. Ferworn, "Intelligent service mesh framework for api security and management," in *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. IEEE, 2019, pp. 0735–0742.

[16] J. Salibindla, "Microservices api security," *International Journal of Engineering Research & Technology*, vol. 7, no. 1, pp. 277–281, 2018.

[17] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv preprint arXiv:1901.03407*, 2019.

[18] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.

[19] N. Moustafa, J. Hu, and J. Slay, "A holistic review of network anomaly detection systems: A comprehensive survey," *Journal of Network and Computer Applications*, vol. 128, pp. 33–55, 2019.

[20] G. Fernandes, J. J. Rodrigues, L. F. Carvalho, J. F. Al-Muhtadi, and M. L. Proença, "A comprehensive survey on network anomaly detection," *Telecommunication Systems*, vol. 70, no. 3, pp. 447–489, 2019.

[21] S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie, "High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning," *Pattern Recognition*, vol. 58, pp. 121–134, 2016.