Open Banking API Security: Anomalous Access Behaviour

1st Dawood Behbehani
School of Mathematics, Computer Science & Engineering
City, University of London
London, United Kingdom
dawood.behbehani@city.ac.uk

2nd Nikos Komninos
School of Mathematics, Computer Science & Engineering
City, University of London
London, United Kingdom
nikos.komninos.1@city.ac.uk

3rd Khalid Al-Begain
Kuwait College of Science & Technology Kuwait
Kuwait
k.albegain@kcst.edu.kw

4th Muttukrishnan Rajarajan
School of Mathematics, Computer Science & Engineering
City, University of London
London, United Kingdom
r.muttukrishnan@city.ac.uk

***Abstract*—Third-party providers pose a significant risk for financial institutions owing to the manner in which banks expose their API to the public. Such threats include inadequate authentication, malicious injections, unsecure key handling. Therefore, financial institutions must adopt a series of countermeasures to mitigate threats exposed by third-party providers, and anomaly detection is considered one such method. In this paper, we develop random forests and a linear kernel SVM to compare the accuracy of our models in predicting anomalous user access behaviour. A dataset that presents users' access behaviour as a numerical feature, including raw API call graphs, is utilised as a case study in this paper. Our novel approach of identifying a risk score and predicting it with a deep neural network showed a high degree of accuracy when the risk scores were developed as a multi-class classification problem**

***Index Terms*—Open banking API security, Open Banking API Anomaly, Open banking API risks**

I. Introduction

The banking sector has been advancing its open banking services significantly to foster competition and innovation. This is achieved by enabling third-party providers (TPPs) to interface with the banks' API services with the aim of providing additional services such as personal budgeting, expenditure analysis, and instant loan origination. However, financial institutions are prone to significant risk on account of the exposure of API services to TPPs. There is no single method for securing API services. Rather, a series of security controls and measures must be adopted in order to secure the API services, such as secure-by-design API development, authentication, and authorization; infrastructure security such as web application firewalls and intrusion prevention systems; and operations and governance such as API policies and monitoring of anomalous behaviour. Therefore, developing robust API mechanisms and detecting API anomalous behaviour are becoming essential to distinguish legitimate and genuine API services from malicious ones. The primary motivation of this paper is to address the problem of identifying anomalous API network traffic.

A. Contributions

- Although previous works have used machine learning models for credit risk assessment, intrusion prevention prediction, and phishing detection, we introduce for the first time a deep neural network model that can be used to predict the risk scores in API security.
- We will also explore the impact of generating continuous and discrete threat scores and expanding datasets using techniques defined in the literature relating to accuracy score.
- We have also highlighted a novel way of looking at API security datasets by reducing the dimensionality (number of features) using PCA analysis.

II. Related work

Machine learning (ML) models have previously been proposed for anomaly detection in various fields like credit risk assessment [1], malware attack [2] [3], and cybersecurity analysis [4]. In credit risk assessment [4], we evaluated a large volume of historical data (job status, credit history, personal account status, etc.) to identify good and bad borrowers. Three ML models were compared [5] [6] [7] using a random forest method (accuracy 69.8%), Choquet integral-based combination classifier (accuracy 64.84%), and decision trees (accuracy 68.4%). Malware attack detection [2] [3] was carried out using support vector structures (SVMs), naive Bayes classifiers, and the k-nearest neighbors algorithm, with SVMs generating better F scores over multiple datasets. However, little research exists on using ML models to predict anomalous behaviour in API security. The problem is also exaggerated by the fact that very few datasets are published online. Previously, [8] developed a framework to detect anomalies in network traffic

using time series forecasting models (SARIMA, LSTM) with a TP (true positive) rate of 14.5% and FP (false positive) rate of 1.7%. Another study using SVMs showed an equally accurate prediction using SVMs [9]. However, both studies do not show us the data they use in their papers. In [10], SVMs are used to classify abnormal API traffic. They used a linear kernel and developed their own dataset on which to train their model. The techniques to create the data have been discussed, but the data has not been provided.

III. Methodology

In this research, based on the literature review, SVMs and random forest methods for developing two or more class classification problems is used. For predicting normal and outlier behaviour, we will develop both random forests and a linear kernel SVM to compare the accuracy of our models. Training will be done over 75% of the total dataset, and testing performed over the remaining 25% of the dataset.

A. Developing the dataset

The literature does not provide sufficient datasets that could be used as a starting point for our machine learning algorithm. A dataset available on kaggle [11] would be used as a starting point (N=1700) . [10] shows how to develop your synthetic dataset. The dataset used in this research records patterns of API usage through the measurement of behavior metrics. User interactions with the API are monitored and transformed into call graphs, which are then distilled into quantitative measures. Essentially, this process converts a series of categorical entities into numerical indicators. The different features (bandwidth consumption, client IP address, number of consecutive requests, connection duration, HTTPS requests) are sampled through a Gaussian distribution and used to train the data over a 2-class classifier (normal and abnormal behaviour) . Baye mentions that if the number of input datasets is less than 100k, a stochastic gradient classifier should be used, and if the number of datasets is more than 100k, an SVM can be used. In this study, we will use the dataset provided by [11], which has 1700 entries and (if needed) generate an extended dataset using the techniques mentioned in [10].

B. Risk score calculation

To develop risk scores, we can use the technique mentioned in [10]. The outlier and normal behaviour classes can be further expanded to give the notion of a risk score. This is helpful since it gives an estimation of the threat level of any user activity. The calculation will work as follows:
We will separate the columns labelled “normal” and “outlier” into scores ranging from 1 to 10. Risk scores 1-5 are classified as normal, and scores 6-10 are classified as outliers. We will use four distributions to calculate the risk score: two discrete and two continuous distributions. The discrete distributions will be used to train a multi-class neural network, whereas the continuous distributions will be used to train a regression-based neural network. More details on how each score is calculated with the predictions can be found in Section 4.

C. Neural network for risk score prediction

We have trained a deep neural network using Principal Component Analysis (PCA) to predict risk scores from our input dataset. PCA has been instrumental in managing the increasing source dataset by effectively decreasing its dimensionality. The PCA identifies the most significant features in a multidimensional dataset, allowing us to generate a new dataset that retains vital information but with fewer dimensions. This assists in managing the complexity of the dataset, improving computational efficiency and reducing the risk of overfitting. Therefore, we have created a modified dataset with new features that are a function of older features. This way, one can easily increase their dataset quantity while accounting for correlations between input features. After PCA, the new dataset would be fed into a feed-forward neural network with N-nodes and M-hidden layers. The output layer will consist of multiple classes (n nodes for an n-class classification and 1 node for regression). The aim of the optimization problem is to identify N and M, which generate the highest F-scores (a statistical measure of the accuracy of ML classifiers) or R-squared scores. As , 75% of the data would be used for training and 25% for testing.

D. Developing ML algorithms (SVM and random forest)

We initiated the process of creating ML frameworks by exploring the provided dataset for anomalies like null rows or extremely large numbers, ensuring such flawed data is removed from our ML algorithm. Before analyzing the results from the ML model, we look at the correlation matrix. A correlation matrix is a table that shows the level of correlation between different variables. In the next part of the project, we would use such an analysis to identify strongly correlated and uncorrelated features. Strongly correlated features tend to produce the same output as their values change. Next, we develop the linear SVM model and random forest model. Support vector machines are supervised learning models that can perform both classification and regression. In essence, the model works by identifying a line/hyperplane that separates the data into the different possible classes. For our system, with only two classes, the SVM model will try to identify a line that separates the multidimensional data into two classes. On the other hand, random forest models also function as a supervised machine learning algorithm, an ensemble learning method that uses multiple decision trees to make predictions. Decision trees are another ML model that uses a tree-like model of decisions to estimate possible consequences and make predictions. We start by separating the dataset into training and testing datasets to train the dataset. A training dataset would be used to train the two ML models, whereas a testing dataset would be used to determine the accuracy of the ML models in

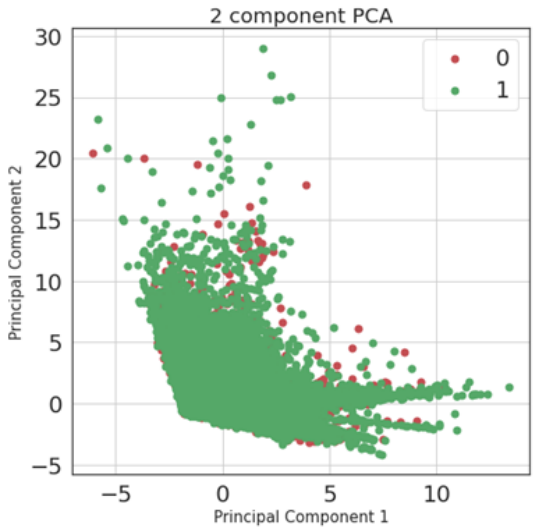

Fig. 1. PCA was performed with two principal components to solve the binary classification problem. The segregation of data is poor

making predictions. Our initial tests identified that the ML models are extremely accurate in predicting normal and outlier behaviours. The SVM model had an F-score of 0.937, whereas the random forest model had an F-score of 0.9727. When 50% and 30% of the data was used as training data, the SVM accuracy was 0.9054 and 0.9036, respectively, but the random forest model still gave an accuracy of 0.967 and 0.9750, respectively.

E. Principal component analysis

PCA (principal component analysis) is a technique that is generally applied to a high dimensional dataset:

1) It aims to reduce the total number of dimensions of the feature set;
2) The new feature set consists of features that are relatively independent of each other.

Figure 1 plots the data from PCA using two principal components when the classification is done over behaviour type (with values 0 and 1). It is easy to see that two principal components are not able to separate the dataset clearly and as such we need more features (more principal components).

Next, we ran an algorithm that runs the PCA algorithm by assuming only a single principal component and increasing the number of principal components (until they equal the number of original dimensions of the feature set). For each step, we calculate the estimated variance that is explained by the given number of principal components. The results are shown in Figure 6. Almost 98% of the variance in data can be explained by five principal components, and as such we will reduce the dimensions of the original dataset from 10 to 5.

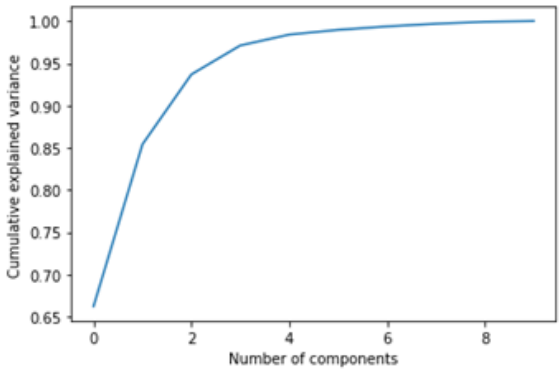

Fig. 2. PCA analysis. Expected variance increased as we increased the number of principal components used.

F. Developing score values

We developed four methodologies to generate the value of our feature 'score':

1) Anytime the behaviour type is 0 (normal), a random number is generated with mean 2.5 and variance 2.5. Anytime the behaviour type is 1 (outlier), a random number is generated with mean 7.5 and variance 2.5. This number is the score.
2) Anytime the behaviour type is 0 (normal), a random number is generated from a uniform distribution between (0,5). Anytime the behaviour type is 1 (outlier), a random number is generated from a uniform distribution between (5,10). This number is the score.
3) Anytime the behaviour type is 0 (normal), a random integer is chosen from 1,2,3,4,5. Anytime the behaviour type is 1 (outlier), a random integer is chosen from 6,7,8,9,10. This number is the score.
4) Anytime the behaviour type is 0 (normal), a random number is chosen from the list $\{0.5,1,1.5,2,2.5,3,3.5,4,4.5,5\}$. Anytime the behaviour type is 1 (outlier), a random integer is chosen from the list $\{5.5,6,6.5,7,7.5,8,8.5,9,9.5,10\}$, This number is the score.

   It should be noted that to make predictions for scores according to (a) and (b), a regression-based deep neural network will be developed. To make predictions for scores according to (c) and (d), a multi-classification neural network will be developed.

The majority of the data follows a power law distribution, i.e. where the frequency of occurrence of a few events is quite likely and the frequency of occurrence of a large number of events is extremely unlikely; 'inter _api _access _duration(sec)', 'sequence _length(count)', 'vsession _duration', 'num _sessions', and 'num _users' follow this power law distribution. 'inter _access _uniqueness' does not follow a familiar distribution. 'num _unique _apis' follows a Poisson distribution ('ip _type'), and 'source' has a binomial

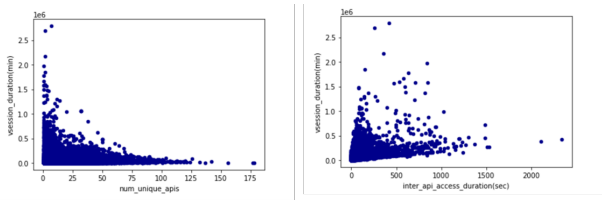

Fig. 3. Scatter plot between 'vsession_duration(min)' and 'inter_api_access_duration(sec)'

Type of neural network	F1/R2 score
Regression with uniform distribution	0.62
Regression with normal distribution	0.61
10 class classification	0.65
20 class classification	0.67

TABLE I
NEURAL NETWORKS AND THEIR F1-SCORES.

distribution. Figure 7 shows 'score', where score is generated according to the uniform distribution discussed above.

Figure 3 shows the scatter plot between 'vsession_duration(min)-nu_unique_apis' and 'vsession_duration(min)-inter_api_access_duration(sec)'.

G. Expanding the dataset

The initial dataset contains 34,423 entries. Predicting the sufficient data quantity for regression-solving deep neural networks is challenging, so we'll increase our dataset tenfold, following a specific expansion protocol:

1) For each feature, estimate the frequency distribution of occurrence of parameters;
2) Normalise the frequency distribution by the total number of counts to generate a probability distribution;
3) For each new row, generate a random number from this probability distribution.

IV. Results and Discussion

Upon running simulation with the 10× expanded dataset, we observed that the maximum achievable F1/R2 score obtained was less than 50% for each of our four different scenarios. We kept on reducing the dataset before coming to realise that the highest accuracy was obtained only with the original dataset.

We therefore used the original dataset to generate the F1/R2 scores. The F1/R2 scores obtained for the four deep neural networks developed are shown in Table 1.

For the 10-class classification problem, we used seven hidden layers with 128 nodes per layer and a ReLU activation function. The batch size before updating the network parameters was 256, and 100 epochs were performed. The F1-score obtained was 0.65 (65% accuracy). We also performed an analysis to identify how far the predicted scores deviated from the actual scores. The results for the 10-class and 20-class neural networks are shown in Tables 1 and 2. For the 10-class classification problem, 73.2% of the predicted score values lie within a margin of two units, and 90.6% of the predicted values lie within a margin of three units. For the 20-class classification problem, 66% of the predicted score values lie within a margin of two units, and 82.5% of the predicted score values lie within a margin of three units. Although the 20-class classification algorithm was found to generate the highest accuracy (F1 score), the 10-class classification model had a higher accuracy within a smaller margin.

V. Conclusion

To conclude, it is essential to mitigate threats exposed by third-party providers when exposing API services publically. In this paper, we developed random forests and a linear kernel SVM to compare the accuracy of our models to predict anomalous user access behaviour. Our work shows that deep neural networks are efficient at both predicting whether the access behaviour is normal or abnormal. Our novel approach of identifying a risk score and predicting it using a deep neural network also showed a high degree of accuracy when the risk scores were developed as a multi-class classification problem.

References

[1] V. Moscato, A. Picariello, and G. Sperlí, "A benchmark of machine learning approaches for credit score prediction," *Expert Systems with Applications*, vol. 165, p. 113986, 3 2021.

[2] M. Barat, D. B. Prelipcean, and D. T. Gavrilut, "An automatic updating perceptron-based system for malware detection," *Proceedings - 15th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2013*, pp. 303–307, 2013.

[3] S. Sheen, R. Anitha, and V. Natarajan, "Android based malware detection using a multifeature collaborative decision fusion approach," *Neurocomputing*, vol. 151, no. P2, pp. 905–912, 3 2015.

[4] J. Martínez Torres, C. Iglesias Comesaña, and P. J. García-Nieto, "Review: machine learning techniques applied to cybersecurity," *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 10, pp. 2823–2836, 10 2019. [Online]. Available: https://link.springer.com/article/10.1007/s13042-018-00906-1

[5] M. Malekipirbazari and V. Aksakalli, "Risk assessment in social lending via random forests," *Expert Systems with Applications*, vol. 42, no. 10, pp. 4621–4631, 6 2015.

[6] A. Namvar and M. Naderpour, "Handling uncertainty in social lending credit risk prediction with a Choquet fuzzy integral model," *IEEE International Conference on Fuzzy Systems*, vol. 2018-July, 10 2018.

[7] J. Sun, J. Lang, H. Fujita, and H. Li, "Imbalanced enterprise credit evaluation with DTE-SBD: Decision tree ensemble based on SMOTE and bagging with differentiated sampling rates," *Information Sciences*, vol. 425, pp. 76–91, 1 2018.

[8] H. Homayouni, I. Ray, S. Ghosh, S. Gondalia, and M. G. Kahn, "Anomaly Detection in COVID-19 Time-Series Data," *SN Computer Science*, vol. 2, no. 4, pp. 1–17, 7 2021. [Online]. Available: https://link.springer.com/article/10.1007/s42979-021-00658-w

[9] G. Yan, "Network Anomaly Traffic Detection Method Based on Support Vector Machine," *Proceedings - 2016 International Conference on Smart City and Systems Engineering, ICSCSE 2016*, pp. 3–6, 1 2017.

[10] G. Baye, F. Hussain, A. Oracevic, R. Hussain, and S. M. Ahsan Kazmi, "API Security in Large Enterprises: Leveraging Machine Learning for Anomaly Detection," *2021 International Symposium on Networks, Computers and Communications, ISNCC 2021*, 2021.

[11] "API security: Access behavior anomaly dataset — Kaggle." [Online]. Available: https://www.kaggle.com/datasets/tangodelta/api-access-behaviour-anomaly-dataset