**International Journal of Communication Networks and Information Security**
2021, 13(1)
ISSN: 2073-607X, 2076-0930
https://ijcnis.org/ **Research Article**

# Advanced Threat Detection in API Security: Leveraging Machine Learning Algorithms

**Piyush Ranjan**[1],
Technology Architect, Cognizant
piyush.ranjan@ieee.org

**Sumit Dahiya**[2],
Solution Architect, Barclays

**ARTICLE INFO**

Received: 05 Jan 2021
Accepted: 06 Feb 2021

**ABSTRACT**

The use of APIs in the current software environments has amplified the vulnerability of APIs in the face of cybsecurity risks. Since APIs are in-between interfaces between the systems, it becomes a common approach for attackers to leverage the available flaws and weaknesses in them. Anti-virus softwares such as firewalls and signature IDS fail to counter enemy easy-to-modify and more complex API attacks. This paper aims to focus on how Machine Learning (ML) algorithms can be used to identify and prevent advanced threats in API security. To address the challenges involved in cyber threat detection, we have developed a general framework combining supervised and unsupervised with targeted reinforcement learning for anomaly, intrusion, injection attacks, DDoS and authentication bypass. Let me analyze some of the machine learning algorithms that can be applied and applied to enhancethe real-time threat detection process for cloud computing networks, including decision trees, random forests, artificial neural networks and models based on deep learning. Finally, we talk about the problems of applying ML models for API security, for example, how to deal with high dimensional data, how to minimize false positive results, and how to protect from newly emerged threats. In order to assess our approach, we used publicly available datasets and simulated API traffic, thus minimizing the false negative rate and enhancing the detection of zero-day exploits. The present findings also show the usefulness of additional machine learning approaches for predictive and responsive API protection over traditional procedures. Additionally, issues concerning dataset quality, feature extraction, and real-time deployment are raised in explaining ways through which ML-based threat detection systems can only be successful. Last, we offer suggestions for future research to advance API security in distributed systems through federated learning and to incorporate explainability in detection methods using xAI.

**Keywords**: API Security, Machine Learning, Anomaly Detection, API Threats, Cybersecurity, Deep Learning.

## 1. Introduction

Application Programming Interfaces, commonly abbreviated as APIs, have been widely used in modern applications because of the communication and data sharing that exists between various applications and services. That is why, with cloud computing environments, microservices architectural models, and IoT devices, the number of vulnerable external APIs is skyrocketing. As more than 80% of the total Web traffic

is nowadays API based, which establishes the importance of safe and reliable API security paradigms. [1-4] API security can be defined as the security of APIs in order to enforce restrictions to only allow specific interactions to authorized persons with the framework endpoints. Conventional API protection involves simple forms of authentication, such as API keys and OAuth tokens and network security includes firewalls. However, such mechanisms always fail to capture new attacks, particularly those that involve new unknown vulnerabilities and/or advanced evasion techniques.

### 1.1. The Emergence of Advanced API Threats

#### 1.1.1. Understanding API Vulnerabilities

Since APIs are now at the core of modern software development, they have become the primary focus of the attackers. Due to the rising adoption of APIs for data sharing, service orchestration and reuse of functionality, API security has emerged as a booming area. The general API vulnerabilities are the same as those which are common to other applications; they include poor authentication, weak rate limiting, and exposure of data. For example, APIs can be programmed incorrectly, thus giving unauthorized people access to some important resources; hence, they can be exploited or the data in them can be leaked. Furthermore, the nature of APIs that are constantly evolving and complex in structure, the cycle of changes is also quick, and such new interfaces contain new vulnerabilities before security specialists even address those threats.

**Figure 1: The Emergence of Advanced API Threats**

#### 1.1.2. Evolution of Attack Techniques

Thus, based on the signs of progress in technology and threat tactics, API threats have become more refined as well. New threats have emerged as a result of the growth of APIs and are directed towards these targets: SQL injection and cross-site scripting (XSS). Nowadays, threat actors use scripts and automated tools to take advantage of openings on a massive scale, and move through API scraping, token stolen, and the abuse of authentication services. They have now turned to methods where they gain long-term access to a system and steal information in bits over time; they are known as advanced persistent threats. Furthermore, new kinds of attacks, such as distributed denial of service (DDoS) attacks in the form of API flood attacks where a large amount of traffic is directed towards the APIs of services, thus making them unavailable.

#### 1.1.3. The Role of Automation and Bots

Automation is a fundamental reason for the development of new forms of complex API threats. Bots can be used for brute force scanning and testing of API for vulnerabilities which is faster and more intensive as a result of utilization by attackers. These bots can be programmed to search out for vulnerable endpoints and, at the same time, take advantage of them hence posing a problem for orthodox security solutions to handle. In dealing with automated attack tools, the use of machine learning also enables bots to learn how they are going to be attacked and what defenses they are going to face so as to adapt to other better ways.

#### 1.1.4. Impact of Cloud and Microservices Architecture

The development of new technologies such as cloud computing and microservices has made more difficult the API security scenario. Although these technologies provide probability, choice and malleability, they have new points of vulnerability. By design, microservices work in an interconnected system; therefore, a gap or crack in one is likely to impact others in a chain-like manner. Notably, the use of third-party APIs raises supply chain risks wherein the weaknesses in the third-party services can lead to the architecting of further risks to the said organization.

#### 1.1.5. Regulatory and Compliance Challenges

There has been growing awareness of API security vulnerability and, thus, the emergence of more regulations. The GDPR and CCPA are among the examples of regulations that impose strict security standards on how personal information has to be processed. Non-compliance carries penalties and other virile consequences on the structures and reputation of organizations. Nevertheless, many companies fail to exert appropriate API security measures that correspond to these compliance requirements, especially when striving to maintain a steady API release rate.

#### 1.1.6. The Need for Enhanced Security Measures

To counter the new advanced API threats, organizations need to implement more effective levels of security. This includes, for instance, having effective mechanisms for controlling the user's identity, like OAuth 2. 0 and JSON Web Tokens (JWT), along with many other measures, such as rate-limiting and throttling, to prevent misuse. Some of the protection measures include; performing the security evaluation and penetration testing at a regular interval to address any loopholes that could be exploited.

Moreover, the capability of integrating a machine learning algorithm and anomaly detection system can further strengthen the threatening alerts so that organizations can quickly manage potential threats.

### 1.2. The Role of Machine Learning in Enhancing Security

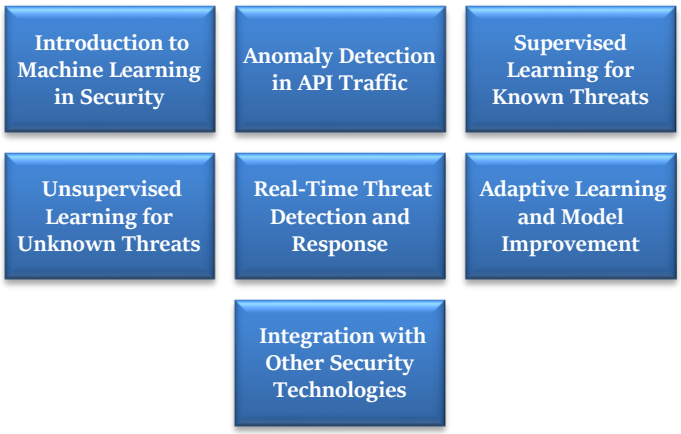



**Figure 1: The Role of Machine Learning in Enhancing Security**

- **Introduction to Machine Learning in Security:** Machine learning (ML) has impacted many sectors, and its use in cybersecurity has slowly started to grow. Through the machining process of learning and making predictions from a set of data, ML makes it possible for security systems to modify and develop to address emerging threats. This capability is particularly valuable in API security; the type of attacks is constantly evolving, requiring adaptive and smart countermeasures. As a tool that is capable of analyzing millions of data and coming up with patterns and results in real-time, it forms a strong foundation for security threat identification and prevention.
- **Anomaly Detection in API Traffic:** Anomaly detection is one of the most traditional ways of using machine learning in API security. In the past, the solutions of security patterns used algorithms xlim and static signatures, which are inefficient in the present threat scenery. But for, ML algorithms, they can learn the normal behaviours of API traffic so that they can know when the traffic steps outside of the normal to flag malicious activities. For example, large increases from one IP address or customers' multiple accesses from one site could be the basis for alert generation and subsequent action on possible threats.
- **Supervised Learning for Known Threats:** To detect previously known risks in API security, various supervised learning approaches are applied. In this approach, models are developed through the use of a data set that has examples of normal and abnormal traffic. Concrete examples of classifiers include Decision Trees, Support Vector Machines (SVM), and Random Forest that enable requests arriving at the API to be filtered based on the patterns noted earlier. This method is especially pertinent for identifying known attack types; thus, organizations may minimize the threats arising from known weaknesses. The problem emerges in the ability to locate large and varied labeled datasets to ensure that the models can generalize.
- **Unsupervised Learning for Unknown Threats:** However, in unsupervised learning models, it is easy to detect unknown threats or zero-day vulnerabilities. What is important about these models is that they do not work with labeled data; they only analyze the traffic and define the traffic that does not conform to the normal pattern as a threat, even if the exact nature of the threat is unknown. One can use such techniques as clustering and autoencoders to reveal new attack vectors due to their ability to detect unusual patterns. For instance, if an API has received requests that are not characteristic of normal traffic, an unsupervised model will mark it as malicious. This capability is important in organizations that are likely to deal with emergent threats which may not be easily detected using conventional procedures.
- **Real-Time Threat Detection and Response:** Another crucial benefit of applying ML in API protection is the possibility of identifying the threats and responding to them in real-time. In addition, they can analyze the traffic data instantaneously as it happens in the APIs as the ML algorithms are capable of detecting immediate and significant activity anomalies. This real-time capacity enables security teams to move with a lot of agility and prevent further damage before an attack occurs. Furthermore, there are other options for automating the response mechanism,

where the system immediately responds to suspicious requests by blocking or throttling them according to set parameters so as to ease the workload on the security personnel.

- **Adaptive Learning and Model Improvement:** Other desirable properties of machine learning systems are also present, namely those of being able to learn from experience and improve themselves. These models are capable of retraining with new data which makes them update themselves with new threats and new patterns of utilization. This continuous learning process helps improve the effectiveness of threat detection to accommodate the various changes that may be present in the environment. There must be feedback loops that help organizations use the outcomes of security incidents to make the models used in training more refined in the protection of APIs against vulnerability.
- **Integration with Other Security Technologies:** Lastly, machine learning can easily be incorporated with any other security solutions, hence improving the security solutions' overall security position. For instance, ML algorithms can work alongside traditional IDPS and firewalls so as to enhance layers of scrutiny as well as intelligence. This paper looked at how correlating data in an organization gives an enhanced view of the security environment; this will help an organization to prevent vulnerabilities that it did not observe or failed to predict due to a lack of a big-picture security view.

## 2. Literature Survey

### 2.1. Traditional API Security Mechanisms

Historically, securing APIs has been accomplished with traditional security measures that include HTTPS Encryption, API keys such as OAuth and JWT, Rate limiting, and Input Validation. [5-8] A situation where these fundamental steps are applicable helps in avoiding various threats to APIs. For instance, HTTPS guarantees the privacy and integrity of data transmitted through the network; API tokens take care of user identification and their rights. However observation of these traditional methods demonstrates its limitation. Even though rate limiting would help against brute force attacks by restricting the number of requests that can be made within a specific period, it is powerless against XSS or SQL injection attacks because such attacks take advantage of the structure of the API and how it handles its inputs. In addition, conventional network defense mechanisms such as firewalls and signature-based IDS offer a form of security to some extent because they are rule/signature-based. It can, therefore, lead to low coverage of attack vectors, including zero-day attacks that are new and unaddressed vulnerabilities.

### 2.2. The Rise of Anomaly Detection

Over the past few years, anomaly detection has become a crucial avenue of research in improving the security of an API through the use of ML. Anomaly detection, in this case, aims at detecting trends that diverge from the typical standardized behavior of the API in the traffic stream to allow for early threat detection. These deviations are ideal to be detected using machine learning models to make them useful for security purposes. I have studied the condition of clusters and their characteristics like K-Means to explore detecting steps of anomalies in API traffic. These studies demonstrate how machine learning algorithms are able to ingest large volumes of traffic data and learn the 'normal' patterns of API use, and in doing so, mark out and quarantine any traffic which looks like it might be suspicious. While the rule-based systems are rigid and fixed in nature, the anomaly detection models offer flexibility in terms of model deployment, which helps organizations become more security-conscious.

### 2.3. Deep Learning in API Security

Newer developments in deep learning have boosted API security even further and given the tools for advanced threat detection. Kinds of deep learning models such as CNNs and RNNs can capture successive phases of time syntax to analyze time series in an API system to single out complex patterns of use suggestive of an attack. In a recent study, we have found the ability of Long Short-Term Memory (LSTM) networks for the prediction of API-based DDoS attacks. Their work also showed that the proposed LSTM model, after training it with traffic data extricated from the year 2018, was able to distinguish between temporal demands of buses with an accuracy of over 90 percent. This high level of accuracy is due to LSTM's ability to learn from sequential API requests over time and is far more effective at detecting slowly evolving complex attacks. This high level of detection, in addition to the benefits afforded to deep learning, does not only improve an environment's security but also the flexibility to counter changing attack patterns.

### 2.4. Supervised vs. Unsupervised Learning for Threat Detection

As far as API threat detection is concerned, at some point there was some debate on the reliability of either supervised or unsupervised machine learning models. Decision tree and Support Vector Machine (SVM), which are supervised learning models, have been chosen effectively for the identification of known threats as such models are trained with previous attack data where requests could be described as either malicious or legitimate intent. However these models face some disadvantages when encountering the zero-day exploit, which does not contain any labeled data and thus is unknown. On the other hand, the unsupervised learning models are more suitable for discovering new threats as it has the capability to find out that some of the API traffic is malicious; while training the model, the traffic was not labeled as malicious. Among the machine learning algorithms that are of help in the identification of outliers include clustering & Auto encoders, which contribute to expressing a higher advantage in the actual RT API data streams. Wang et al. have also revealed that the use of unsupervised learning provides the capability of learning new ways of attack since a new kind of threat model of attack is always emerging.

### 2.5. Challenges in Machine Learning-Based API Security

The potential benefits of machine learning in marketing API security have been discussed above, but certain obstacles need to be overcome to one day fully realize its advantages. Among the challenges, one main challenge is Feature Engineering where one has to decide and extract appropriate data elements that would feed into the model. Feature engineering is a crucial aspect of making sure that models correctly identify revenue and attacks by distinguishing normal traffic from malicious ones. Insufficient feature selection can result in low model performance and, subsequently, efficient threat identification. Moreover, the problem of false positive case detections still poses a significant threat. Sensitive models could mark legitimate requests as threats, thus interrupting APIs when there is no actual threat present or when user experiences could be deprived. Additionally, given that API threats are an active space and continuously change, this is also another challenge; attackers start using new techniques, meaning that features and machine learning models used need constant updates and training. There will always be additional costs to introduce changes as new risks appear. As new usage patterns emerge in the API traffic, and that is why organizations have to develop good practices regarding model maintenance to ensure that their security mechanisms are adjusting to the new threats.

## 3. Methodology

### 3.1. System Architecture for Threat Detection

The proposed architecture for machine learning (ML)-based API threat detection is divided into three essential components: These are the Data Collection Module, the Feature Extraction and Preprocessing, and the core ML Models. Each of these plays an important part in protecting APIs from advanced threats and making certain the system is able to quickly identify both known and unknown attack patterns. [9-14]This makes it possible to mount security in tiers so that each new threat can be managed as it develops.

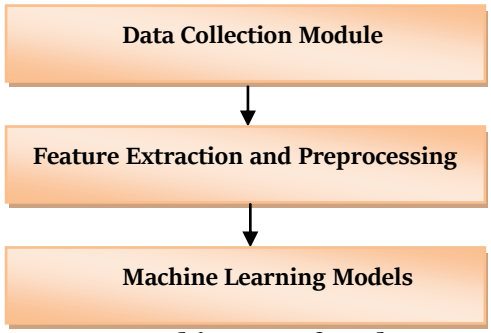



**Figure 3: System Architecture for Threat Detection**

- **Data Collection Module:** The Data Collection Module is the one that is charged with feeding real-time traffic data of its API from several endpoints. This encompasses request and response headers, the actual payload data, time stamps, user credentials, IP addresses, and the user device profile. The fundamental goal is to record as many interactions between clients and APIs as possible. These data are collected continuously, which builds the background of threat detection. The question is how to gather this information while not trading off performance to an intolerable degree or adding noticeable latency, which is paramount to keeping API operations smooth. The high quality and the completeness of the dataset minimize the possibility of errors in the consequent threat detection steps.
- **Feature Extraction and Preprocessing**: After obtaining the raw API traffic data the next operation that takes place is Feature Extraction and Preprocessing. It is important for this phase

because there is always noise or unnecessary data included in raw data, which reduces the performance of machine learning models. Data cleaning helps in discarding noisy, less corrupt records from the dataset. Normalization is used to bring the data to a universal format so that it matches your data submission requirements, such as the volume of traffic and the size of requests and responses. Specifically, access patterns regarding API endpoints, user agent strings, IP location, request size, and time intervals between requests are extracted. They are instrumental in allowing the detection of 'anomalous' behavior on otherwise legitimate APIs. In a nutshell, useful preprocessing means the data is put in the most favorable form of readiness as it can capture the model predictions.

- **Machine Learning Models:** The core of the threat detection architecture is based on the Machine Learning Models where different algorithms investigate the processed data in order to identify an anomaly or an attack. To accommodate a wide variety of attacks and their levels, the architecture includes Supervised Learning, Unsupervised Learning, and Deep Learning models.
- **Supervised Learning Models**: Classification techniques consisting of Decision Trees, Random Forests, and Support Vector Machines (SVM) are applied to detect the required patterns of attack. The type of data sets usually used in training such models, however, goes beyond directly identifying APIs; they have to be safe or unsafe, in the broadest sense of the word. While supervised learning is quite efficient in exactly identifying known threats, which are pre-classified by nature, including potentially hostile SQL queries or cross-site scripting attacks, it is not as accurate when it comes to the categorical identification of other types of intrusions. Thus, these models, which leverage history to detect instances quite rapidly and accurately, apply only where the threats have observable signatures.
- **Unsupervised Learning Models**: Concerning the unfamiliar threats/novelties forecasting, the learning technique used comprises K-Means Clustering,Auto encoders and Gaussian Mixture Models (GMM). The models require no initial labels on the data and are useful in identifying new or developing attack forms. Security specialists analyze API traffic to find features that do not correlate with the norm, where, in fact, they do not even pay attention to new threats. Perhaps the only argument against it is that by virtue of grouping similar data points and by producing outliers, unsupervised learning is a moving defense against a more sophisticated attacker who may use different tactics.
- **Deep Learning Models:** Specifically, LSTM and CNN approaches are used for classifying slow, enduring, continued invasions. They are ideal for time series data; thus, suitable for detecting shifts in API utilization over time, such as unsteady DDoS attacks. CNNs can be used as models for the feature extraction process in higher levels of data and identify specific possible key features that may not be otherwise recognizable. Some of these deep learning models enhance the adversary's detection of not only some novel breaches but also some futuristic attacks, this makes them fairly significant for the current API protection.

### 3.2. Model Training and Testing

This paper elucidates how training and testing of the machine learning models in API threat detection plays a critical role in arriving at a reliable and effective system. It also prevents the models from introducing any extraneous noise that otherwise would have affected the generalizability of the models in detecting malicious patterns with minimal false alarms using new unseen data. In the next subsections, we expand on the major frame of the process of model training and testing, which encompasses data partitioning, model training, cross-validation, model testing, and performance fine-tuning.

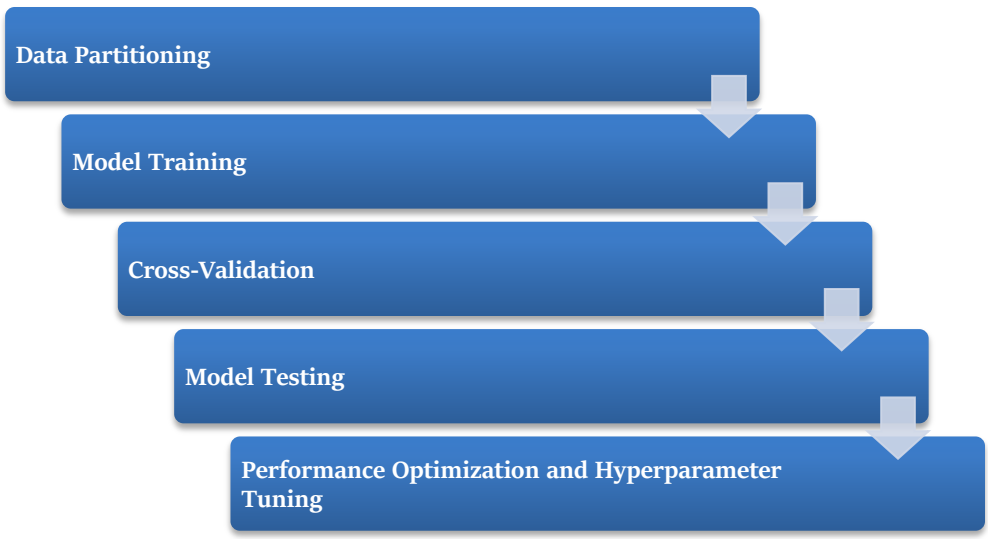


**Figure 4: Model Training and Testing**

- **Data Partitioning:** Data partitioning is the first process of building machine learning models. We have the API Security Dataset available on Kaggle that features a diverse set of API traffic samples including both safe and unsafe calls. This dataset was divided into two primary subsets: the Training Set 70 percent and the Testing Set 30%. The training set is used to acquaint the models with the pattern of the normal and the malicious API traffic and has plenty of samples available for use. The testing set, on the other hand, is used to assess the model at a later stage after training and develop confidence in whether the model can generalize well when applied to new data. The division 70/30 is adopted by machine learning commonly to ensure they have enough data for training but still have a solid structure on how to test models' effectiveness.
- **Model Training:** During the model training step, machine learning algorithms are at work with the labelled training set to analyze which calls made to API, the size of the payload and the user agent string correlates to which outcome, and these could either be benign or malicious. The models, therefore, master the relations between the aforementioned input features and their labels.
- **Cross-Validation:** To check overfitting, a case where a model will perform well for the sample dataset but poorly for other different datasets, we used cross-validation methods. That is because complex models tend to produce high Bias and possibly high Variance in the training data set; thus, they are not ideal for inferential use. In the present study, K times cross-validation has been used where the total samples of training data are partitioned into K equal samples. These subsets are used for testing purposes, while the other K-1 can be used for training purposes. This operation is repeated K times so that each segment of data should be used both for training and for testing. As will be observed, the averaging provides the right estimation of the models' performance as compared to the mere average values. Apart from helping reduce overfitting, cross-validation has a critical role in preventing the entire model from being influenced by some form of data split within the application environment.
- **Model Testing:** After training, we proceeded to test the model using the previously reserved testing data set, which has 30% of all the data. This set includes API traffic data that the model had not seen during training to provide for an independent assessment of the performance metrics. During testing, the model classifies each API request as either benign or malicious, and these classifications are compared to the actual classifications that are in the test data set. Testing on the model is important to determine how well the algorithms in machine learning are able to detect the anomalous traffic coming from the API while not flagging legitimate traffic as malicious. Thus, we calculate accuracy, precision, recall, F1 score, as well as false positive rate in order to evaluate the possibility of the model to monitor API threats in real-time efficiently.
- **Performance Optimization and Hyperparameter Tuning:** For model optimization, therefore, hyperparameter tuning was done, which is an important step in training machine learning models. Hyperparameters are certain parameters special to each type of model (for example, the learning rate in the neural network or the number of trees in random forests) which may cause a huge difference in the effectiveness of the model. Tuning them enhances the goodness of the model and the ability to generalize on unseen data. In our work, we employed methods like grid search and random search in order to determine the best hyperparameter combination. Cross-validation methods systematically search different parameter tuning to arrive at the best performance on the testing set. Therefore, tuning these hyperparameters helped us to drastically improve the model while searching for API threats in order to create a more reliable security system.

### 3.3. Evaluation Metrics

Following training, the models were tested using various test assessments that sought to measure the capacity of the models in the identification of API threats. [15,16] These are important metrics needed to measure how effectively the models can identify anomalous behaviour without creating high levels of false alarms.

- **Accuracy**: Accuracy shows the ratio of total correct predictions of both benign and malicious requests to the total set of predicted frames. It serves as a general measure of the model's accuracy. However, it can be deceptive in imbalanced datasets thus where the amount of benign traffic is significantly higher than the malicious traffic. High accuracy suggests that the model is

capable of distinguishing between a normal and an attack; however, it is riskful to use this single metric to test the model's robustness.

- **Precision**: The best way to measure precision is the ratio of actual instances of malicious requests predicted by the model out of all the instances of requests the model deemed as malicious (true positives plus false positives). This explains why high subscripts of precision show a high ability of the model to reduce false positives, which, in other words, shows that most of the API requests that the model flags are malicious. This is important in preventive API security since too many false positives only lead to service interferences, creating the false impression that a threat was detected.
- **Recall**: Recall, sometimes called sensitivity, is the ratio of the number of true evil requests to the total number of requests that were actually evil. It quantifies the model's aptitude for identifying all the potential threats that may occur. The recall is valuable as it captures all possible security threats; however, setting a high recall level without adequate precision will result in too many false positives, creating alerts or service interruption.
- **F1 Score**: The F1 score of a model can be obtained by taking the mean of precision and recall's harmonic mean. In situations where both false positives as well as false negatives are undesirable, this metric comes in handy. A higher F1 score of the model means that the majority of threats are being detected while false alarms remain low.
- **False Positive Rate (FPR):** FPR stands for the fraction of benign API requests which got misclassified by the model as a malicious one. It is apparent that low laboratory false positives are highly important to prevent the majority of actual traffic from being flagged. In the API security context, high FPRs are equally undesirable, as they cause interruption of service for legitimate use or subpar quality of experience for the end users. Therefore, achieving a state where FPR is low but recall does not suffer is one of the biggest concerns in designing efficient threat detection solutions

## 4. Results and Discussion

### 4.1. Model Performance Analysis

When implementing and evaluating different machine learning models for API threat detection, we were able to determine major differences in their efficiency based on accuracy and false positive statistics. It is, therefore, important for decision-makers to obtain the necessary understanding of these differences in order to be able to select the right model for certain operational scenarios.

**Table 1: Model Performance Metrics**

| Model | Accuracy (%) | Precision (%) | Recall (%) | F1 Score | False Positive Rate (%) |
|---|---|---|---|---|---|
| Decision Tree | 85.4 | 82.1 | 80.5 | 81.3 | 15.7 |
| Random Forest | 91.3 | 89.9 | 88.0 | 88.7 | 10.4 |
| Support Vector Machine | 90.0 | 87.9 | 86.5 | 87.2 | 12.0 |
| LSTM | 95.6 | 94.5 | 93.8 | 94.1 | 7.3 |

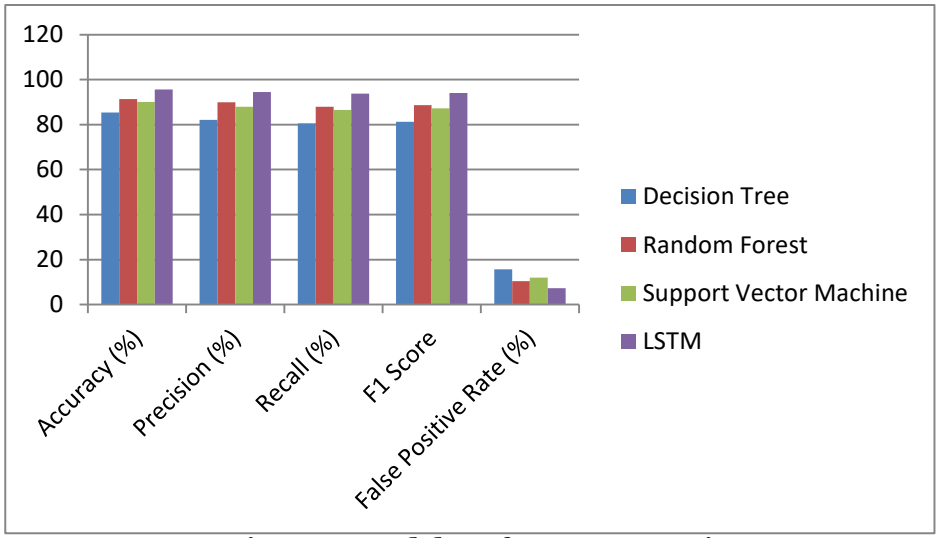



**Figure 5: Model Performance Metrics**

### 4.1.1. Discussion of Results

Random Decision Tree followed this at a detection accuracy of 87.1%, Naïve Bayes with the lowest detection accuracy of 74.1% and LSTM with the highest of all 95.6%. This outstanding performance can be attributed to LSTM's architecture, that well captures temporal dependencies in API traffic data. While the static models cannot analyze the sequence of data over time, the LSTMs are logically suited for identifying slow, progressive and continuous attacks like DDoS attacks. Such attacks are usually relatively sustained and not as dramatic as a sudden blip, and this makes LSTMs pick up on such activity when other models fail.

Different machine learning models, like Decision Trees and Random Forests, were less accurate than the NLP model they only attained 91.3% in the Random Forest. Even if they cannot learn complex attack patterns, he noted that simple models have their advantages in terms of computational speed. Due to their relatively less computation-intensive nature, they can easily be deployed in organizations experiencing resource constraints to continue running efficient security operations without over-stretching their resources.

### 4.1.2.Computational Complexity

However, there are problems with computational complexity in LSTM networks; otherwise, they deliver outstanding results, usually in cases concerning limited resources. The problem seems to arise from LSTM's architecture,which requires a lot of power and memory when computing, a factor that slows down API response time. This latency could, however, be a challenge particularly for high throughput systems where end users will require rates of response in good service delivery.

LSTM models can yield higher performance than CNN models, albeit due to computational complexity involving state information across sequences, the models may take longer time to process. This goes against conventional architectures like Random Forests and Support Vector Machines (SVM) even though they can have a deeper structure training and inference time is comparatively shorter.

### 4.1.3. Trade-offs in Model Selection

From these factors, we have the following considerations when choosing the API threat detection model: precision to the time and effort exhausted.

- **For High-Stakes Environments:** However, when it comes to performing the role of detection with high accuracy, LSTM networks can be applied in some cases, for example, in financial organizations or in critical infrastructure, where it is allowed to use more resources and computer time. Another solution that organizations may invest in is optimized hardware or cloud services to do away with latency.
- **For Resource-Constrained Settings**: Whereas we may have settings in which resources are even more limited; maybe then the likes of Random Forest or even SVM may come in handy. These models are arguments for reasonable reliability to facilitate API security in real-time without much resource and time to analyze them.

## 4.2. False Positives and Avoidance

### 4.2.1. Overview of False Positives

As with all models in our analysis, one of the issues observed in the current models was a relatively high false positive rate. False positive is a scenario whereby normal API calls are flagged as containing malicious traffic hence generating an alert while causing an interruption of other necessary services. This is especially undesirable in the API security domains since the constant availability of services is a crucial factor for their quality and system stability.

### 4.2.2. Discussion of False Positives

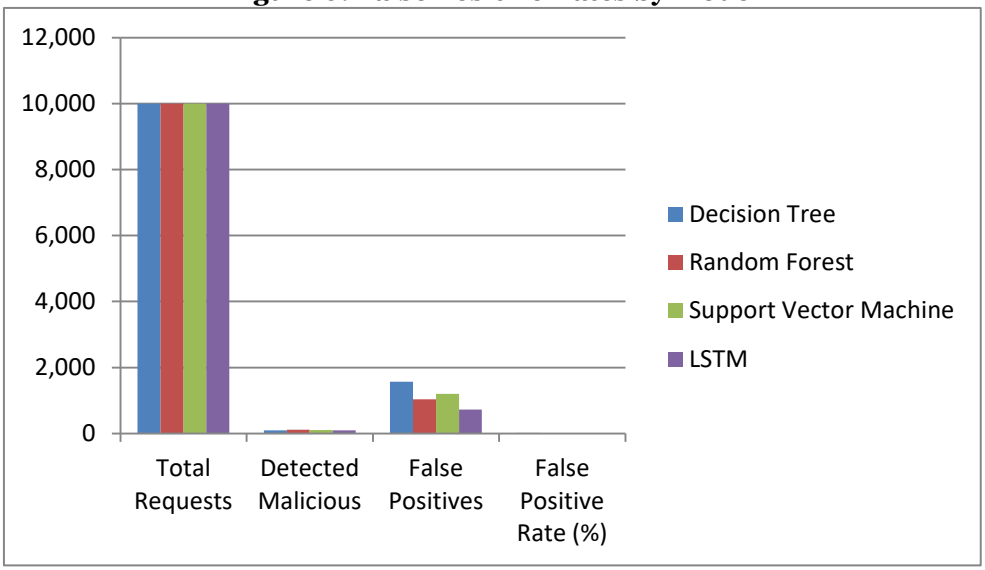
The false positive rate is markedly higher for some models than others, thus highlighting the problem. While comparing the metrics of the models, 'False Positive' was highest in the Decision Tree model, at 15.7%, incorporating the tendency of DT to label legitimate requests as threats. However, the LSTM model that we deployed, which gave the overall highest accuracy, had a false positive rate of 7.3%.

False positive rates this high can interfere with legitimate service provision by degrading user experience, perhaps causing customers to lose trust in the service provider. For example, users may receive undesirable blocks or delays in service when their API calls are mistakenly put in the wrong category. This can lead to dissatisfaction, and there will be an indication that people will have to look for other service providers.

**Table 2: False Positive Rates by Model**

| Model | Total Requests | Detected Malicious | False Positives | False Positive Rate (%) |
|---|---|---|---|---|
| Decision Tree | 10,000 | 100 | 1570 | 15.7 |
| Random Forest | 10,000 | 120 | 1040 | 10.4 |
| Support Vector Machine | 10,000 | 110 | 1200 | 12.0 |
| LSTM | 10,000 | 95 | 730 | 7.3 |

**Figure 6: False Positive Rates by Model**




### 4.2.3. Adaptive Threshold Mechanism

To mitigate this important problem of false positives, we incorporated a new strategy of using dynamic thresholds to control the sensitivity of the models. This adaptive approach allows us to react to dynamic traffic distribution and operating parameters efficiently.

### 4.2.4. Key Benefits of the Adaptive Mechanism:

- **Reduced False Positives**: Thus, during periods with higher historical false positive rates, it is possible to reduce the sensitivity of the model so as not to disrupt legitimate traffic unnecessarily. This change assists in guaranteeing that real user orders are handled efficiently while at the same time retaining some measure of protection.
- **Improved Detection**: On the other hand, we can increase the sensitivity of the model when threat awareness is particularly high. This change makes it possible to improve the system sensitivity while achieving a relatively low false positive rate. For example, if the system has patterns of a known form of attack, raising the sensitivity may likely allow the capture of other forms of threats that may prevail but be unnoticed.

The problem of high FPR is most often encountered in ADS specifications, including those of API security. Nevertheless, all the same, with the right tuning and correct introduction of adaptive mechanisms, the problem can be decreased to a considerable extent. With this, we are able to achieve sensitivity adjustment with the help of historical data for preventing API traffic from being flagged while still maintaining strong threat detection.From here, it remains imperative to pay constant attention and adjust these mechanisms in the future because the nature of API traffic and risks could change over time. For future research, the identification of even more accurate models could be a direction, as well as identifying further ways of mitigating the false positive rate. Finally, it is essential to strike the right balance between improving the detection accuracy of API vulnerabilities and reducing the impact on legitimate services to obtain effective API security.

## 5. Conclusion

This paper focuses on the high value that is achievable when applying machine learning for the identification of advanced threats in API security to protect significant digital assets. With APIs being the backbone of current and future applications, security has become more important than ever. Our study shows that the use of machine learning with a focus on deep learning use cases such as LSTM achieves better accuracy in reading multilayer attack patterns, including DDoS and API. As LSTMs are capable of identifying temporal patterns in API traffic, they can distinguish between minor traffic changes over time, which makes them ideal for detecting new threats that other models may overlook. This capability is important as attackers are always evolving their tactics, requiring similar levels of counter-tactics.

However, the study shows that deep learning models based on images are highly accurate; there is a significant issue that when generating false positives, it is detrimental to the user experience. Solving this problem is critical for practical application in real contexts where it is necessary to balance between protection and convenience. Adaptive threshold mechanisms show one of the ways through which possible high false positive rates can be addressed while maintaining high threat detection rates. This way, making sensitivity levels more dynamic and adjusting them to previous activity data can help improve API security while keeping users satisfied.

Future research must focus on using explainable artificial intelligence (XAI) approaches within machine learning models for API security. With the advancement in artificial intelligence and machine learning models, the demands for interpreting the result of the models increase. This is why Explainable AI is used to explain the thought processes by which a certain result is arrived at so that the stakeholders can be trusted with the automatic systems currently in use. This pertinence is especially marked in Security Computing contexts, where organizations need to explain their defense measures and give the rationale behind identified threats. Firstly, when applied to detection mechanisms, the proposed approach increases interpretability and allows organizations to better trust existing security automation as well as manage interactions between human analysts and outputs from machine learning systems.

Further, examining federated learning raises the potential for future research direction of distributed API security. Through Federated learning, several organizations can jointly train their machine learning models without the need to share their data. Such an approach enables richer model performance, which is necessary for modern applications while ensuring that data remains private and secure. This way, through federated learning, organizations can utilize the ideas that are collected across various datasets, which will help enhance security against complex threats. The future of API security must depend on the adaptation of advanced Machine learning techniques, Explainable AI, and Federated learning techniques that will help to design betterprotected API security.

## 6. References

1. Siriwardena, P. (2014). Advanced API Security. Apress: New York, NY, USA.
2. Koloveas, P., Chantzios, T., Alevizopoulou, S., Skiadopoulos, S., & Tryfonopoulos, C. (2021). intime: A machine learning-based framework for gathering and leveraging web data to cyber-threat intelligence. Electronics, 10(7), 818.
3. Qolomany, B., Al-Fuqaha, A., Gupta, A., Benhaddou, D., Alwajidi, S., Qadir, J., & Fong, A. C. (2019). Leveraging machine learning and big data for smart buildings: A comprehensive survey. IEEE Access, 7, 90316-90356.
4. Wang, A., Machida, Y., deSouza, P., Mora, S., Duhl, T., Hudda, N., ... & Ratti, C. (2023). Leveraging machine learning algorithms to advance low-cost air sensor calibration in stationary and mobile settings. Atmospheric Environment, 301, 119692.
5. Xue, M., Yuan, C., Wu, H., Zhang, Y., & Liu, W. (2020). Machine learning security: Threats, countermeasures, and evaluations. IEEE Access, 8, 74720-74742.

6. Mamidi, S. R. (2024). The Role of AI and Machine Learning in Enhancing Cloud Security. Journal of Artificial Intelligence General Science (JAIGS) ISSN: 3006-4023, 3(1), 403-417.
7. Prasad, N. R., Almanza-Garcia, S., & Lu, T. T. (2010). Anomaly detection. Computers, Materials, & Continua, 14(1), 1-22.
8. Datta, J., Dasgupta, R., Dasgupta, S., & Reddy, K. R. (2021, September). Real-time threat detection in ueba using unsupervised learning algorithms. In 2021 5th International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech) (pp. 1-6). IEEE.
9. Aldairi, M., Karimi, L., & Joshi, J. (2019, July). A trust-aware unsupervised learning approach for insider threat detection. In 2019 IEEE 20th International conference on information reuse and integration for data science (IRI) (pp. 89-98). IEEE.
10. Schiefelbein, M. C. (2008, May). Information architecture for threat detection systems. In 2008 IEEE Conference on Technologies for Homeland Security (pp. 589-592). IEEE.
11. Lobato, A. G. P., Lopez, M. A., Sanz, I. J., Cardenas, A. A., Duarte, O. C. M., & Pujolle, G. (2018, May). An adaptive real-time architecture for zero-day threat detection. In 2018 IEEE International Conference on Communications (ICC) (pp. 1-6). IEEE.
12. Ayoade, G., Akbar, K. A., Sahoo, P., Gao, Y., Agarwal, A., Jee, K., ... & Singhal, A. (2020, June). Evolving advanced persistent threat detection using provenance graph and metric learning. In 2020 IEEE Conference on Communications and Network Security (CNS) (pp. 1-9). IEEE.
13. Legg, P. A., Buckley, O., Goldsmith, M., & Creese, S. (2015). Automated insider threat detection system using user and role-based profile assessment. IEEE Systems Journal, 11(2), 503-512.
14. Almousa, M., Basavaraju, S., & Anwar, M. (2021, December). Api-based ransomware detection using machine learning-based threat detection models. In 2021 18th International Conference on Privacy, Security and Trust (PST) (pp. 1-7). IEEE.
15. Sharma, I., Kaur, A., Kaushik, K., & Chhabra, G. (2023, July). Machine Learning-Based Detection of API Security Attacks. In International Conference on Data Science and Applications (pp. 285-297). Singapore: Springer Nature Singapore.
16. Guan, Z., Bian, L., Shang, T., & Liu, J. (2018, August). When machine learning meets security issues: A survey. In 2018 IEEE International Conference on Intelligence and Safety for Robotics (ISR) (pp. 158-165). IEEE.
17. Shatnawi, A. S., Jaradat, A., Yaseen, T. B., Taqieddin, E., Al-Ayyoub, M., & Mustafa, D. (2022). An android malware detection leveraging machine learning. Wireless Communications and Mobile Computing, 2022(1), 1830201.
18. Singh, J., & Singh, J. (2022). Assessment of supervised machine learning algorithms using dynamic API calls for malware detection. International Journal of Computers and Applications, 44(3), 270-277.
19. Sagar, R., Jhaveri, R., & Borrego, C. (2020). Applications in security and evasions in machine learning: a survey. Electronics, 9(1), 97.
20. Gao, J., Li, L., Kong, P., Bissyandé, T. F., & Klein, J. (2019). Understanding the evolution of Android app vulnerabilities. IEEE Transactions on Reliability, 70(1), 212-230.