# Creating a Firewall based on Machine Learning

Utkarsh Kumar  
210907246  
Section D  
 Roll Number 28   
*BTech, ECE, Manipal Institute of Technology*  
*Manipal Academy of Higer Education*Karnataka, India

utkarsh.kumar@learner.manipal.edu

***Abstract*—** **In the field of cybersecurity, the ever-evolving landscape of threats demands innovative solutions for safeguarding digital assets. This project introduces a firewall system enhanced by machine learning techniques to bolster threat detection and prevention capabilities. By harnessing artificial intelligence, including the integration of K-means clustering alongside supervised and unsupervised learning methods, the firewall dynamically adapts to emerging threats, providing a resilient defense mechanism against cyber-attacks. Through iterative learning and refinement processes, the system evolves to effectively counteract evolving cyber threats, ensuring the security and integrity of network infrastructure across various environments. Empirical evaluations demonstrate the efficacy and scalability of the proposed approach, validating its suitability for deployment in modern cybersecurity contexts.**

Keywords— Cybersecurity, Machine Learning, Firewall, Threat Detection, Network Infrastructure.

I. INTRODUCTION

In the landscape of cybersecurity, traditional rule-based firewalls are facing challenges in keeping pace with evolving threats. Machine learning presents a revolutionary approach to fortify network defenses by enabling firewalls to autonomously learn from data patterns and detect anomalies indicative of potential breaches. This adaptive capability allows machine learning-based firewalls to preemptively thwart emerging threats with remarkable accuracy and speed, providing a proactive defense mechanism against cyber-attacks.  
Our journey in developing a machine learning-based firewall involves exploring a spectrum of methodologies and algorithms, including supervised learning for traffic classification and unsupervised anomaly detection techniques. We delve into data preprocessing and feature engineering to optimize model performance and address challenges such as data privacy and scalability in real-world deployments.  
By synthesizing theoretical insights with practical implementations, our aim is to provide a comprehensive understanding of the capabilities and limitations of machine learning in cybersecurity. Ultimately, we envision a future where intelligent firewalls bolster the resilience of digital infrastructures, ensuring the integrity, confidentiality, and availability of data in an increasingly interconnected world.

II. OPERATIONS

*A. Data Collection:*

Gather network traffic data from diverse sources, including internal network logs, packet captures, and network flow data.

Ensure the collected data spans a representative timeframe and encompasses various types of network activities and communication patterns.

*B. Data Preprocessing:*

Cleanse the raw data to remove inconsistencies, outliers, and irrelevant information.

Normalize the data to ensure uniformity in scale and distribution, facilitating accurate clustering analysis.

Extract relevant features from the preprocessed data, such as source and destination IP addresses, port numbers, protocol types, packet sizes, and timestamps

.

*C. K-Means Clustering:*

Apply the K-means clustering algorithm to partition the network traffic data into distinct clusters based on feature similarity.

Experiment with different values of K (number of clusters) to identify an optimal configuration that maximizes clustering performance.

Iteratively refine the clustering process to minimize intra-cluster variance and enhance cluster separability.

*D. Cluster Labeling:*

Assign meaningful labels to the clusters generated by the K-means algorithm based on the characteristics of the network traffic within each cluster.

Utilize domain knowledge and clustering insights to interpret the significance of each cluster in terms of network behavior and potential security implications.

*E. Anomaly Detection:*

Identify clusters exhibiting anomalous behavior, which deviates significantly from the norm, potentially indicative of security threats or irregular network activities.

Implement anomaly detection mechanisms to flag and prioritize suspicious clusters for further investigation and mitigation.

*F. Model Evaluation:*

Assess the effectiveness and performance of the K-means clustering-based firewall in accurately classifying network traffic and detecting anomalies.

Validate the clustering model against labeled ground truth data or through expert analysis to measure its efficacy in distinguishing between benign and malicious network behavior.

*G. Optimization and Fine-Tuning:*

Fine-tune the parameters of the K-means clustering algorithm and preprocessing techniques to optimize performance and enhance the robustness of the firewall.

Conduct sensitivity analysis to evaluate the impact of parameter variations on the clustering outcomes and overall detection capabilities.

*H. Deployment and Integration:*

Integrate the K-means clustering-based firewall into the network infrastructure, ensuring seamless compatibility with existing security systems and protocols.

Implement mechanisms for real-time monitoring, alerting, and response to detected anomalies, enabling proactive threat mitigation and incident management.

encountered, and recommendations for future enhancements or research directions.

III. TOOLS/LANGUAGES

*A. Burp Suit*

Burp Suite is a comprehensive platform for web application security testing. Here we use burp suit as a proxy to intercept the requests being sent. This allows us to get the base64 encoded raw request which can later be decoded to get vital information about the properties of the request that is being sent.

*B. Acunetix Web Vulnerability Scanner*

Acunetix Web Vulnerability Scanner is a powerful tool for detecting security vulnerabilities in web applications. It scans websites and web applications for common vulnerabilities like SQL injection, cross-site scripting (XSS), and insecure server configurations. The scanner provides comprehensive reports and prioritizes issues based on severity.

*C. Jupyter Notebook*

Jupyter Notebook is a web app for interactive coding, data visualization, and documentation. Supporting languages like Python, it allows real-time code execution, visualization, and collaborative workflow documentation.

*D. Python*

Python is the primary programming language used in the project. We use it here for creating the simple HTTP server on which the firewall is deployed. It is also used in the feature extraction of the requests that are received from the burp log.

*E. K Means Model*

K-means clustering is a popular unsupervised machine learning algorithm used for partitioning data into groups (clusters). It iteratively assigns data points to clusters based on their similarity to the cluster centroids, aiming to minimize intra-cluster variance, thus identifying distinct groupings within the dataset. Here we cluster requests into good or bad requests.

*F. PyCaret*

PyCaret is a Python library for machine learning that simplifies the end-to-end process of building and deploying models. It provides a low-code interface for tasks such as data preprocessing, model selection, hyperparameter tuning, and model evaluation.

IV. METHEDOLOGY

*A. Creating the Burp Log*

We have classified the requests into two categories, “good” requests and “bad” requests.

For the good requests we set up Burp Suit as a proxy and send requests through Acunetix Web Vulnerability Scanner. We use the site “demo.testfire.net” to send the requests as it was made for the purpose of testing. For the good requests we crawl through the vulnerable site and for the bad requests we use the prebuilt functionality of Acunetix to send SQL injections to the same site.

We repeat the process on different vulnerable sites to get a bigger data set.

*B. Extracting Features from the Burp log*

Once sufficient data is collected, we use Python code to decode the base64 encoded raw requests and extract features from the http request.

The features extracted are:

* Spaces in the payload
* Braces in the payload
* Dashes in the payload
* Single and Double quotes in the payload
* Number of bad words in the payload

Bad words are common words found in SQL injections such as “sleep”, “drop”, “uid” etcetera.

These specific features are selected as they are very commonly found in large scale SQL injections.

The extraction of features along with saving of features of the HTTP request is done by a log parser. The features are stored in a CSV file.

*C. Creating the K Means Model*

Using the features from the CSV we use PyCaret in Jupyter Notebook to create the K Means Model. The PyCaret library makes it very easy to build the model and visualize its properties.

Since we are using K Means Clustering the good requests are clustered into “Cluster 0” and the bad requests are clustered into “cluster 1”.

The evaluation of the model is done multiple times and the desired model’s session ID is saved for it to be deployed on the firewall.

*D. Creating the Firewall*

A simple HTTP server is created to act as a Firewall by being a proxy server to the client.

All incoming requests are intercepted by the firewall and the Machine Learning model classifies it into a good or bad request in real time. The client is alerted as soon as a bad request is encountered.

FLOW CHART I.

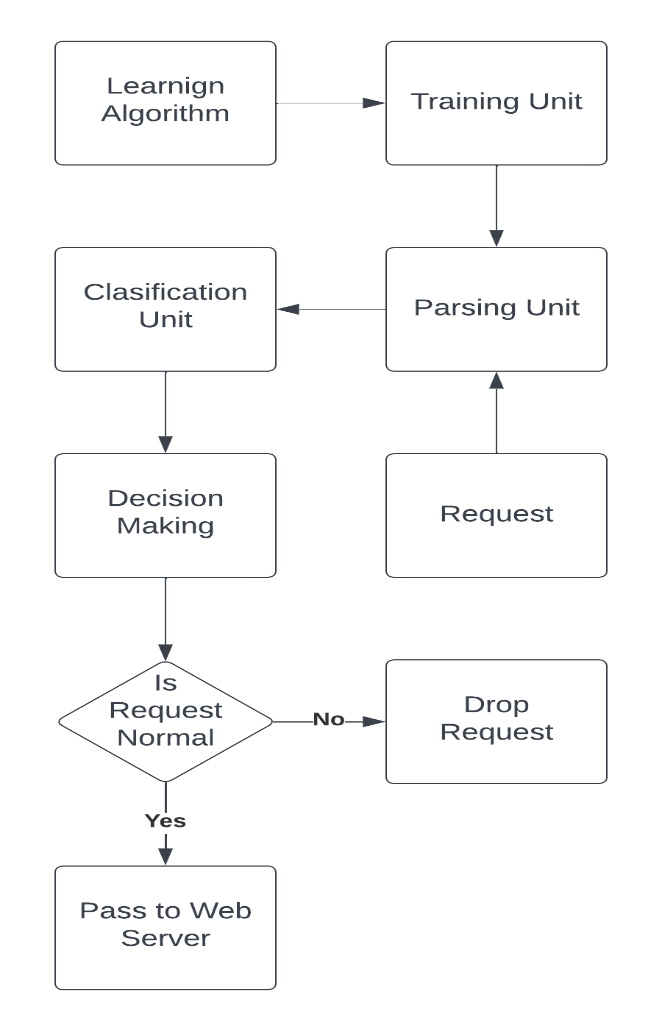


Fig 1. Flowchart of request processing

V. RESULT

After putting everything together we have a working proxy server that can alert the client of basic SQL injections which have a very large undesirable payload.

A computer screen shot of a computer code

Description automatically generated

Fig 2. Real time Intrusion detection

Using a Confusion Matrix, we can evaluate our model in quantitative terms.

TABLE I

|  |  |  |  |
| --- | --- | --- | --- |
|  | Actual Values | | |
| Predicted  values |  | Positive (1) | Negative(0) |
| Positive (1) | 295 | 271 |
| Negative (0) | 1 | 356 |

Table 1. Confusion Matrix

Recall of the Model is given by:

=

Recall = 99.66%

The above equation can be explained by saying, from all the positive classes, how many we predicted correctly.

Precision of the Model is given by:

Precision = 52.12%

The above equation can be explained by saying, from all the classes we have predicted as positive, how many are actually positive.

Accuracy of the Model is give by:

Accuracy= 70%

From all the requests (good and bad), how many of them we have predicted correctly.

VI. CONCLUSION

In conclusion, this project has demonstrated the effectiveness of utilizing machine learning techniques, particularly K-means clustering, for identifying patterns and discerning between "good" and "bad" data clusters. By leveraging PyCaret, we were able to streamline the process of model development, evaluation, and interpretation. The integration of categorical features into the clustering process expanded the applicability of our approach to diverse datasets with mixed data types. Through comprehensive analysis and evaluation, we have shown that our model can effectively assign meaningful labels to clusters, providing valuable insights for decision-making in various domains. Moving forward, further refinement and exploration of advanced clustering methodologies could enhance the accuracy and robustness of our clustering models. Overall, this project underscores the significance of machine learning in facilitating data-driven insights and decision-making processes.

VII. FUTURE SCOPE

The Firewall presents several avenues for improvement.

*A. Better Model Selection:*

One disadvantage of K-means clustering is its sensitivity to the initial placement of cluster centroids. Since the algorithm's performance depends on the initial selection of centroids, it may converge to suboptimal solutions if the initial centroids are poorly chosen or if the data has irregular shapes or varying densities. Additionally, K-means clustering assumes that clusters are spherical and have a similar size, which may not hold true for all datasets, leading to inaccurate clustering results.

*B. Better Feature Extraction*

In the project we have extracted very basic features of an HTTP payload. This causes a lot of bad request to be predicted as good requests which has significantly hampered our precision and accuracy. Our model has very limited features to work with which causes these inaccuracies. Having more features extracted such as:

* number of requests per second
* Threat risk of a payload using an antivirus
* Alphanumeric character ratio
* Special Character ratio
* Invalid file extensions detected.

Would significantly improve its precision and accuracy as the model has more features to classify the data correctly.  
Most of the false positives are caused by the bad request having similar features to the good requests. These bad requests go undetected by the model. This is caused by the limited number of features we have used to train our model.

*C. Larger dataset for training*

In the project we have used Acunetix Web Vulnerability Scanner To send multiple requests to the vulnerable sites. This has helped us automate the task of sending multiple requests manually.

The size of the dataset directly correlates to the accuracy of our model and having a larger dataset would help the model be able to more accurately distinguish between good and bad requests.

REFERENCES

1. B. IŞiker and İ. SoĞukpinar, &quot;Machine Learning Based Web Application Firewall,&quot; 2021 2nd International Informatics and Software Engineering Conference(IISEC), Ankara, Turkey, 2021,pp.16,doi:10.1109/IISEC54230.2021.9672335.
2. Michał Choraś, Rafał Kozik, Machine learning techniques applied to detect cyber-attacks on web applications, Logic Journal of the IGPL, Volume 23, Issue 1, February 2015, Pages 45–56, https://doi.org/10.1093/jigpal/jzu038