A Mini Project Report on

WEB APPLICATION FIREWALL USING MACHINE LEARNING

Submitted in partial fulfillment for the
degree of Bachelor of Technology in
Computer Science and Technology

Submitted by
Aditi Majalkar (32)
Avantika Mane (33)
Shobha Nayak (34)

Under the guidance of
Prof. Sumedh Pundkar

Department of Computer Science and Technology

USHA MITTAL INSTITUTE OF TECHNOLOGY

S.N.D.T WOMEN’S UNIVERSITY

MUMBAI – 400049

2023 - 24

Approval Sheet

This is to certify that Aditi Majalkar, Avantika Mane, Shobha Nayak have completed Mini Project report on the topic "WEB APPLICATION FIREWALL USING MACHINE LEARNING" satisfactorily for the bachelor's degree in computer science and technology under the guidance of Mr. Sumedh Pundkar during the year 2023-24 as prescribed by S.N.D.T Women's University, Mumbai.

Guide	Head of Department
Prof. Sumedh Pundkar	Prof. Kumud Wasnik

Principal
Dr. Yogesh Nerkar

Examiner 1	Examiner 2

Declaration

We declare that this written submission represents our ideas in our own words and where others’ ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will cause disciplinary action by the Institute and can also invoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Aditi Majalkar (32)

Avantika Mane (33)

Shobha Nayak (34)

Date: 18/03/2024

Acknowledgement

We have a great pleasure to express our gratitude to all those who have contributed and motivated during our project work. We thank our principal, Dr. Yogesh Nerkar, our HoD, Prof. Kumud Wasnik and our guide, Prof. Sumedh Pundkar for mentoring us.

Date: 18/03/24

Aditi Majalkar(32)

Avantika Mane(33)

Shobha Nayak(34)

Abstract

This research introduces an innovative approach to enhance internet security through the integration of machine learning into firewall systems. Conventional firewalls which are governed by static rules, are susceptible to evasion by proficient attackers.
Our methodology holds machine learning to enhance the detection of cyber threats, marking a significant advancement in internet security. The machine learning system demonstrates commendable performance in accurately distinguishing between malicious and regular network traffic.
We specifically investigate the application of machine learning in Web Application Fire- walls (WAFs) to tackle the evolving challenges of cybersecurity. While machine-learning- based WAFs exhibit promising capabilities in thwarting injection attacks and providing simplified management, their efficacy against contemporary web application attacks necessitates further scrutiny.
By incorporating machine learning into firewall systems, our approach offers a more dynamic and adaptive defense mechanism, fortifying internet security against sophisticated threats.

Contents

1. Introduction **7**

1.1 Problem Statement ………………………………………………………………. 8

2. Literature Survey

2.1 Technical Papers …..………………………….…………………………………. 9

2.2 Existing Systems …..……………………………………………………………… 11

3. Proposed System

3.1 Workflow of the Project …..……………………….…………………………….. 12

3.2 Clustering Analysis with Kmeans ……………………….……………………….. 13

4. Hardware and Software Requirements

4.1 Hardware Requirements …..……………………………….……………………….. 16

4.2 Software Requirements …..………………………….………………………….. 16

5. Result and Discussion

5.1 Comparison of Models ……..……………………….……………………………. 17

5.2 Implementation

5.2.1 Feature extraction …..……………………………………….……………….. 20

5.2.2 Clustering Analysis ……...…………...………………………….…….…….. 21

5.2.3 Intrusion detection and request Routing ..…………………………………..... 22

5.2.4 Server Initialization and Execution ……………………………...……... 23

5.3 Output …..…………………………………………………………….…..……. 24

6. Application **25**

7. Conclusion

7.1 Future Scope …...……………….……………………………………………..……. 26

7.2 Conclusion …..…………….………..………………………………...……. 26

8. References **27**

Chapter 1

Introduction

In today's dynamic cybersecurity landscape, the traditional paradigms of safeguarding net works through fixed-rule firewalls face mounting challenges. Skilled attackers can easily bypass these rule-based defenses, highlighting the urgent need for innovative security solutions that can adapt to new threats. In response to these challenges, this study introduces a pioneering approach by harnessing the power of machine learning to fortify internet security.

By integrating machine learning into web application firewalls (WAFs), this project aims to revolutionise the efficacy of threat detection and mitigation. While conventional firewalls rely on predetermined rules, they often fall short in identifying sophisticated cyber threats that deviate from known patterns. Unlike traditional firewalls that rely on predefined rules, machine learning algorithms can dynamically analyze and categorize network traffic, making it easier to distinguish between malicious and benign activities.

The project explores the potential of ML-based WAFs in addressing the growing complexity of cyber attacks, with a particular focus on their effectiveness against injection attacks and their ability to simplify administrative management. However, it also acknowledges the need for further research to ensure the reliability of these systems against the full range of contemporary web application threats.

Through the exploration of machine learning in WAFs and the evaluation of their performance against evolving threats, this study aims to make significant contributions to the cybersecurity field. By conducting thorough analysis and empirical testing, it seeks to pave the way for a more resilient and adaptable defense mechanism against cyber threats.

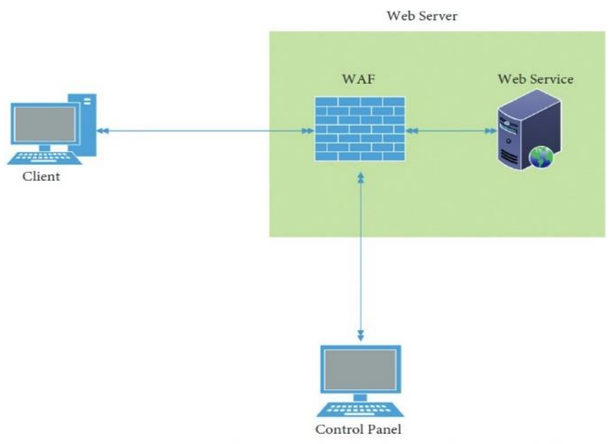


FIGURE 1: Architecture of the web server environment with proposal WAF.

Traditional firewall systems with fixed rules are increasingly vulnerable to sophisticated cyber threats, as skilled attackers can outsmart these rigid defenses.

To address this pressing issue, there is a need for a comprehensive solution that integrates machine learning into firewall systems, aiming to enhance the detection of cyber threats and fortify internet security. The effectiveness of machine-learning-based Web Application Firewalls (WAFs) against evolving web application attacks also requires thorough investigation, considering the dynamic nature of modern cyber threats.

1.1 Problem Statement

Traditional firewall systems with fixed rules are increasingly vulnerable to sophisticated cyber threats, as skilled attackers can outsmart these rigid defenses. To address this pressing issue, there is a need for a comprehensive solution that integrates machine learning into firewall systems, aiming to enhance the detection of cyber threats and fortify internet security. The effectiveness of machine-learning-based Web Application Fire walls (WAFs) against evolving web application attacks also requires thorough investigation, considering the dynamic nature of modern cyber threats

Chapter 2

Literature Survey

2.1 Technical Papers

A thorough literature survey was conducted to gain insights into existing research relevant to our project. A literature survey, also known as a literature review, is a critical aspect of research that involves analyzing and synthesizing previously published work on a specific topic. Its purpose is to identify gaps in knowledge, assess the current state of research, and lay the groundwork for further investigation.

During our research, we referenced four key papers related to the development of Web Application Firewalls. The first paper proposes a WAF system utilizing machine learning algorithms for the detection of common web attacks. The second paper explores the inte- gration of machine learning techniques, such as shallow neural networks and optimizable decision trees, to enhance internet firewall performance by determining actions for each communicated packet.

The third paper discusses signature-based and machine learning-based WAFs, focusing on the detection of SQL injection, cross-site scripting, and cross-site request forgery attacks using artificial neural networks and other machine learning algorithms. Lastly, the fourth paper addresses the need for comprehensive WAF testing tools, an all-in-one WAF testing tool capable of performing multiple testing methods, such as fuzzing, payload execution, bypassing, and foot printing, to ensure the robustness of WAF implementations

.

International Standard Serial No.	**Research Paper Name**	**Year Published**	**Name of the Author(s)**	**Methodologies and Technologies**	**Findings and Remark**
5280-1588	Web Application Firewall Using Machine Learning and Features Engineering	June 2022	Aref Shaheed, M. H. D. Bassam Kurdy	Machine learning algorithm	To detect common web attack
2640-0111	Improving Internet Firewall Using Machine Learning Techniques	November 2023	Martha Ozohu Musa, Temitope Victor-Ime	Shallow neural networks and optimizable decision trees. Integration of machine learning algorithm into IPS	To determine action for each communicated packet. To improve performance of firewall.
1877-0509	Signature based and machine learning based web application firewalls	2021	Simon Applebauma, Tarek Gaberb, Ali Ahmed	Artificial neural network. Machine learning algorithm.	SQL injection detection. cross site scripting, cross site request forgery detection.
1065-1072	Web Application Attacks Detection Using Machine Learning Techniques	December 2018	Gustavo Betarte, Alvaro Pardo Rodrigo MArtinex	Machine learning Data collection and labelling	To use machine learning techniques to reduce false positives generated by WAFs while maintaining high true positive rate.

2.2 Existing System

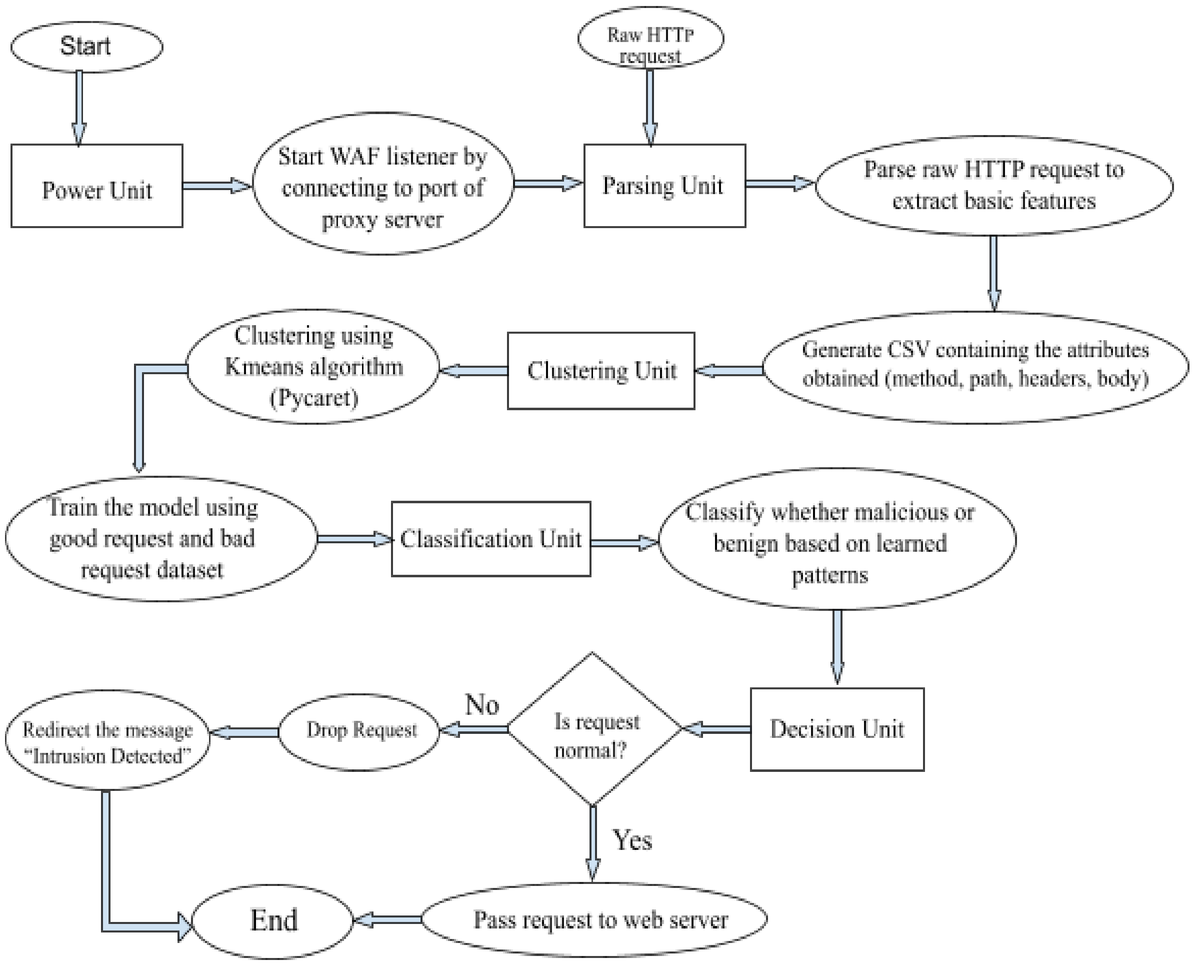
1. **ModSecurity**: ModSecurity is an open-source WAF that incorporates machine learning techniques for threat detection and mitigation. It uses anomaly detection algorithms to identify and block malicious traffic patterns.
2. **Imperva SecureSphere WAF**: Imperva SecureSphere WAF employs machine learning algorithms to analyze web traffic and detect suspicious behavior indicative of cyber threats. It continuously learns from new data to enhance its ability to identify and block attacks.
3. **Signal Sciences**: Signal Sciences offers a WAF solution that utilizes machine learning to provide real-time protection against web application attacks. It analyzes traffic patterns and user behavior to detect and prevent malicious activity.
4. **Cloudflare WAF**: Cloudflare's WAF leverages machine learning algorithms to defend against various web-based attacks, including SQL injection, cross-site scripting (XSS), and distributed denial-of-service (DDoS) attacks. It continuously adapts to evolving threats to ensure effective protection.
5. **F5 Advanced WAF**: F5 Advanced WAF incorporates machine learning capabilities to detect and block sophisticated attacks targeting web applications. It uses behavioral analysis and anomaly detection techniques to identify and mitigate security threats in real-time.
6. **Barracuda WAF**: Barracuda WAF integrates machine learning algorithms to provide comprehensive protection against OWASP top 10 threats and other advanced web application attacks. It offers adaptive security policies based on evolving threat landscapes

Chapter 3

Proposed System

3.1 Workflow of The Project

Our WAF using machine learning starts with the Power Unit, initiating the WAF listenerby connecting to a proxy server port. Raw HTTP requests are intercepted and sent to the Parsing Unit for extracting basic features like method, path, headers, and body, compiled into a CSV file.
In the Clustering Unit, parsed data undergoes Kmeans clustering using Pycaret, training the model with datasets containing good and bad requests. Following training, the model proceeds to the Classification Unit for request classification based on learned patterns.
If a request is flagged as malicious, it enters the Decision Unit, verifying its normalcy.If abnormal, the request is dropped, triggering an ”Intrusion Detected” message redirection. Otherwise, normal requests are passed to the web server for handling.


Start
Raw HTTP request
Power Unit
Start WAF listener by connecting to port of proxy server
Parsing Unit
Parse raw HTTP request to extract basic features
Generate CSV containing the attributes obtained (method, path, headers, body)
Clustering Unit
Clustering using Kmeans algorithm (Pycaret)
Train the model using good request and bad request dataset
Classification Unit
Classify whether malicious or benign based on learned patterns
Decision Unit
Is request normal?
No
Yes
Drop Request
Redirect the message "Intrusion Detected"
Pass request to web server
End

Proposed system architecture of Web Application Firewall using Machine Learning

3.2 Clustering Analysis with Kmeans

Clustering is a type of unsupervised learning technique used to group similar data points together based on their features. It helps in identifying inherent patterns and structures in the data without the need for labeled output.

- **KMeans Algorithm:** KMeans is one of the most commonly used clustering algorithms. It partitions the dataset into 'k' distinct clusters, where each data point belongs to the cluster with the nearest mean value. The algorithm iteratively assigns data points to the nearest centroid and updates the centroids until convergence.

  In our analysis, we utilized the KMeans algorithm to cluster HTTP request data. The features extracted from the HTTP requests included the occurrences of single quotes, double quotes, dashes, braces, spaces, and certain predefined 'badwords' which indicates potential security threats.

- **How KMeans Works**:

  Initialization**:**

  - Start by randomly assigning each data point to an initial group (cluster).
  - Calculate the centroid (centre) for each cluster. The centroid represents the mean of all data points in that cluster.

  Assignment Step:

  - Evaluate each observation and assign it to the closest cluster.
  - Closest is determined by **Euclidean distance** between a data point and a cluster's centroid.

  Update Step**:**

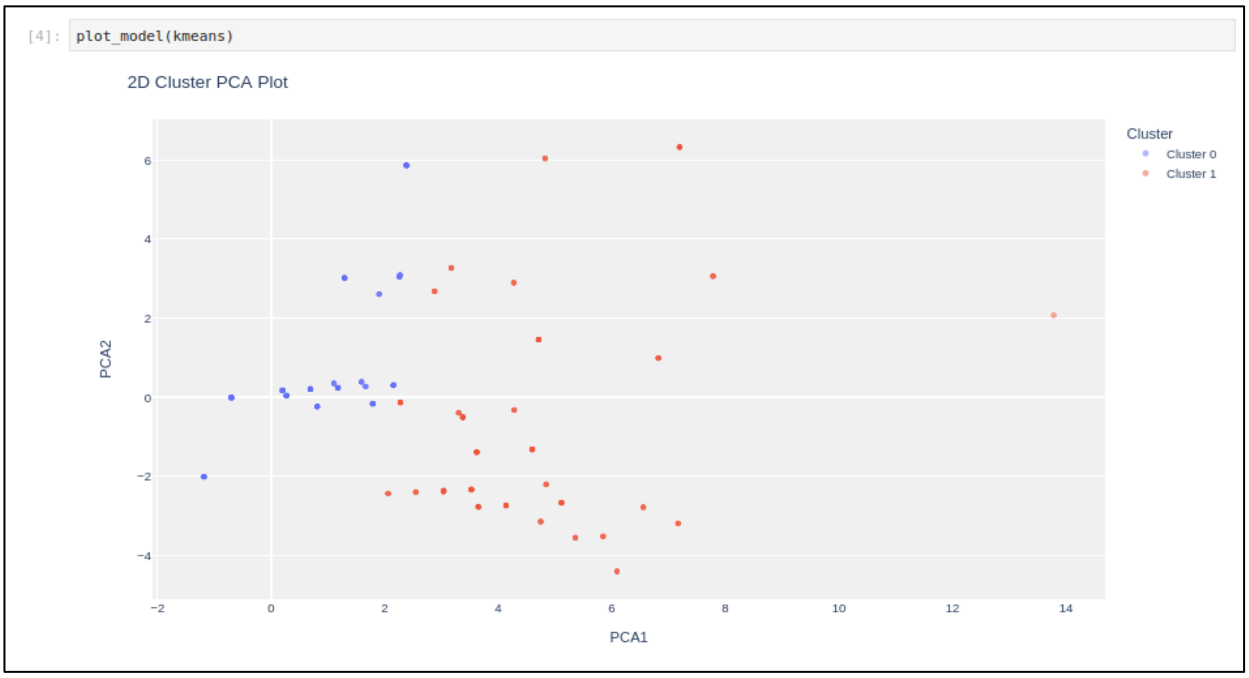
  - When a cluster gains or loses a data point, recalculate its centroid.
  - Repeat the assignment and update steps until no further reassignments occur.

  Result:

  - All clusters have minimum within-cluster variance, keeping them as small as possible.
  - Sets with minimum variance and size have data points that are as similar as possible.

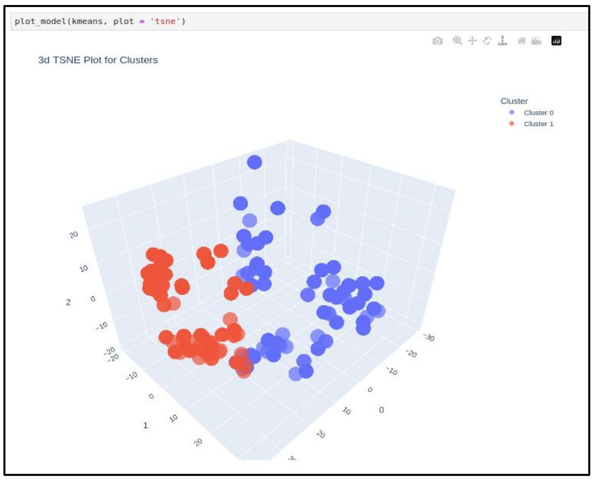
- **Plotting the Model**

  We visualized the clustering results using t-SNE (t-distributed Stochastic Neighbor Embedding) plot. The t-SNE plot provides a three-dimensional representation of the high-dimensional feature space, which enables us to visualize the clusters in a reduced dimensionality. The cluster scatter plot illustrates how the data points are distributed across different clusters based on selected features.

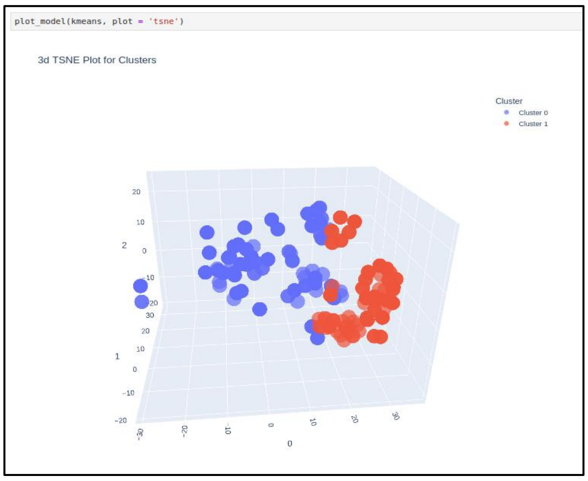


```
[4]: plot_model(kmeans)
```

2-D Plotting of Clusters



```
plot_model(kmeans, plot = 'tsne')
```



```
plot_model(kmeans, plot = 'tsne')
```

3-D Plotting of Cluste

- The plot represents two distinct clusters obtained from KMeans.
- **Cluster 0** is depicted with **red dots**, while **Cluster 1** is represented by **blue dots**.
- The 3D plot shows the distribution of data points across three axes.
- The grid lines help understand the distribution along each axis for better interpretation.

- **Testing and Classification**

  Once the KMeans model was trained on the original dataset, we applied it to predict clusters for a test dataset of HTTP requests. By comparing the predicted clusters with the original clusters, we evaluated the model's performance in classifying new HTTP requests. This allowed us to identify potential security threats and anomalies in real-time web traffic, aiding in the detection and prevention of cyber attacks.

Chapter 4

Hardware And Software Requirements

4.1 Hardware Requirement

1. Processor:
   - Intel Core i3 or higher
   - Minimum Speed: 2.5 GHz
   - AMD Ryzen 3 or higher series
2. RAM:
   - Minimum 6 GB (Recommended)
3. Hard Disk:
   - Minimum 250 GB SSD for efficient data processing and storage
4. Network Interface Card (NIC):
   - Gigabit Ethernet for fast and reliable data transfer

4.2 Software Requirement

1. Operating System:
   - Windows 7 or higher versions can be used
2. Web Browser:
   - Mozilla Firefox
   - Google Chrome
3. Proxy Tool:
   - Burp Suite for capturing and parsing HTTP requests and responses
   - OWASP ZAP scanning tool
4. Programming Language and Frameworks:
   - Python 3.x for scripting and machine learning model integration
5. Integrated Development Environment (IDE):
   - PyCharm or Jupyter Notebook for Python scripting
6. Automation Library:
   - PyCaret for machine learning automation

Chapter 5

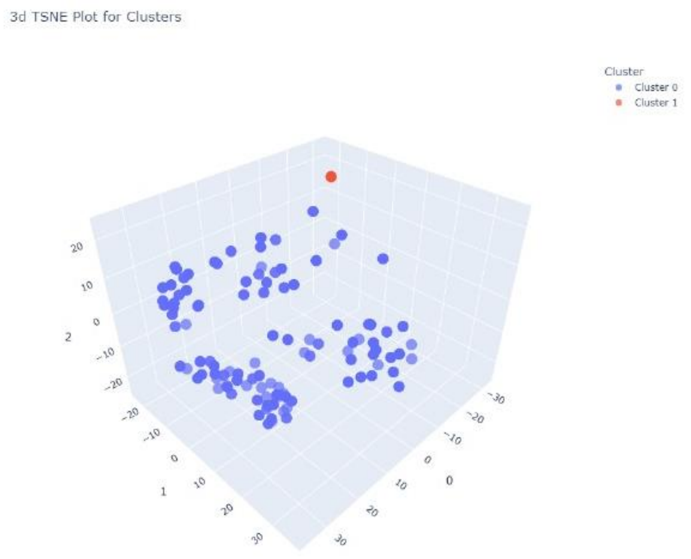
Result And Discussion

5.1 Comparison of Models

In this section, we compare the clustering results obtained from **Hierarchical Clustering**, **DBSCAN**, and **K-means** models. Our dataset comprises a nearly equal number of 'good' and 'bad' requests, which necessitates a balanced clustering approach where each cluster should ideally contain a similar number of data points.

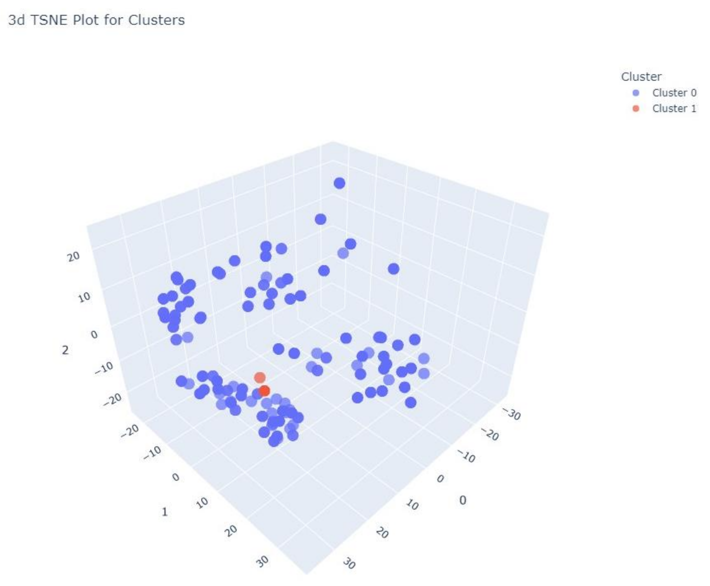
5.1.1 Hierarchical clustering Plot

The 3D plot produced by Hierarchical Clustering revealed an imbalanced distribution of data points across the clusters. One cluster contained the majority of the data points, while the other consisted of only a few. This imbalance is not ideal for our analysis, as it does not reflect the near-equal distribution of good and bad requests in our dataset.



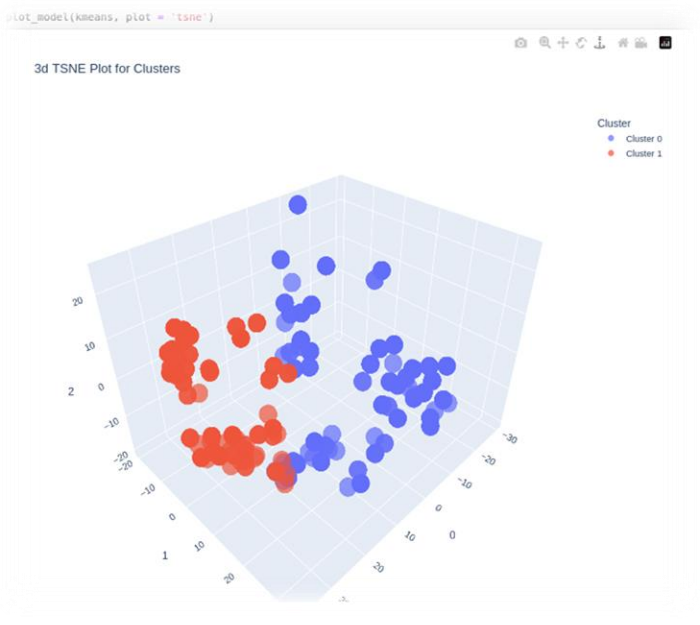
5.1.2 DBSCAN Plot

DBSCAN's plot exhibited a similar issue. The nature of DBSCAN to form clusters based on density and noise led to one large cluster and several smaller or single data point clusters. This resulted in an uneven cluster distribution, which again, is not representative of our dataset's structure.



5.1.3 Kmeans Plot

In contrast, the K-means algorithm provided a more balanced clustering result. The plot showed two clusters with almost equal numbers of data points. This balance is crucial for our project since it aligns with the actual distribution of good and bad requests within our dataset. The equal-sized clusters facilitate a more accurate analysis and interpretation of the data.



```
plot_model(kmeans, plot = 'tsne')
```

Given the requirement for balanced clustering and the comparative analysis of the plots, we decided to proceed with the **K-means** model. Its ability to partition the data into clusters of nearly equal sizes makes it the most suitable choice for our project's objectives. This decision is further supported by the metrics we obtained, which indicate that K-means performed effectively in segregating the data into meaningful clusters.

While each algorithm has its strengths, **K-means** was chosen for its scalability, ease of interpretation, consistency, simplicity, ease of quantitative evaluation, and potential for predictive modeling, making it the most suitable option for our project's requirements.

5.2 Implementation

5.2.1 Feature Extraction

The purpose of the **ExtractFeatures function** is to convert raw HTTP request paths into a structured format that can be used for machine learning analysis. By quantifying certain characteristics of the request paths, the function prepares a dataset that can be used to identify patterns indicative of SQL injection attacks.

Process

The feature extraction process involves the following steps:

1. URL Decoding: The request path is URL-decoded using urllib.parse.unquote. This is necessary because HTTP requests are often URL-encoded, and decoding them reveals the actual characters used in the request.
2. Feature Identification: The function identifies specific features within the request path that are commonly associated with SQL injection attacks. These features include:
    - The number of single quotes ('), which could indicate an attempt to terminate a string literal in SQL.
    - The number of double quotes ("), which could also be used in SQL string literal terminations.
    - The number of dashes (--), which are often used in SQL to comment out the remaining part of a query, potentially neutralizing security checks.
    - The number of parentheses ((), which could be used to alter the precedence of operations in SQL.
    - The number of spaces ( ), which could be used to separate SQL keywords and clauses.
    - The count of predefined “badwords”, which are specific SQL keywords that could be used maliciously to manipulate database queries.
3. Feature Vector Creation: The counts of these features are compiled into a list, which is then converted into a pandas DataFrame. This DataFrame serves as the input for the clustering model, with each column representing a different feature.

Output

The output of the ExtractFeatures function is a pandas DataFrame with a single row containing the counts of each feature identified in the request path. This DataFrame is labeled with the following columns: ['single_q', 'double_q', 'dashes', 'braces', 'spaces', 'badwords'].

Significance of Feature Extraction

The extracted features are significant because they provide measurable data points that can be analysed by the clustering algorithm. By identifying requests with unusually high counts of these features, the system can flag potential SQL injection attempts and prevent them from reaching the database.

5.2.2 Clustering Analysis

The objective of clustering analysis is to categorize HTTP requests into clusters based on their feature vectors. This categorization helps in identifying patterns that may indicate normal or malicious behaviour.

Process

The clustering analysis follows these steps:

1. Data Importation: The HTTP request data is imported from a CSV file using pandas.read_csv. This dataset contains various features of HTTP requests that are potential indicators of SQL injection attacks.
2. Data Preprocessing with PyCaret: The setup function from the pycaret.clustering module is used to preprocess the data. The steps include:
   - Normalization: The data is normalized to ensure that the feature values have the same scale, which is crucial for the performance of many clustering algorithms.
   - Feature Selection: The features to be used for clustering are specified as numeric features, which include counts of single quotes, double quotes, dashes, braces, spaces, and badwords.
   - Feature Ignoring: Certain features that are not relevant to the clustering, such as ‘method’, ‘path’, ‘body’, and ‘class’, are ignored.
3. Model Creation: The create_model function is used to instantiate the K-Means clustering algorithm with the specified number of clusters, which is two in this case. K-Means is a popular clustering algorithm that partitions the data into clusters based on the nearest mean.

Output

The results of the clustering analysis are used to label each HTTP request as belonging to either Cluster 0 or Cluster 1. One cluster represent normal traffic, while the other represent potentially malicious requests indicative of SQL injection attacks.

Significance of Clustering Analysis

Clustering analysis is significant because it allows for the unsupervised categorization of data. In the absence of labeled data indicating which requests are malicious, clustering provides a way to separate requests into groups that can then be analyzed for patterns of normal or abnormal behavior.

5.2.3 Intrusion Detection and Request Routing

The SimpleHTTPProxy class is an integral part of the intrusion detection system. It inherits from SimpleHTTPRequestHandler and is designed to intercept HTTP GET requests, analyze them for potential security threats, and route legitimate requests to their intended destinations.

<u>Class Configuration</u>

- Proxy Routes: The class maintains a dictionary of proxy routes that determine where incoming requests should be forwarded. This allows the proxy server to act as an intermediary, fetching resources on behalf of clients.

<u>Intrusion Detection Mechanism</u>

- GET Request Handling: The do_GET method is the core of the intrusion detection mechanism. It processes incoming GET requests by splitting the URL path and extracting features that are indicative of SQL injection attempts.
- Feature Analysis: The extracted features from the URL path are analyzed using a pre-trained machine learning model. This model predicts whether the request is normal or potentially malicious.
- Intrusion Alert: If the model identifies a request as potentially malicious (Cluster 1), the system logs an intrusion alert. This serves as a warning that the request may be part of an SQL injection attack.

<u>Request Routing</u>

- Proxy Request Forwarding: For non-malicious requests, the proxy_request method forwards the request to the appropriate destination based on the proxy routes. This method handles the communication with the target server and relays the response back to the client.
- Error Handling: In case of an HTTP error during the forwarding process, the method captures the error code and sends an appropriate response to the client, ensuring that the error is handled gracefully.

In essence, the output of the SimpleHTTPProxy class includes the detection of potential security threats and the seamless routing of client requests to their respective destinations. It ensures the security and efficiency of the network traffic it handles.

5.2.4 Server Initialization and Execution

Initialization

- **Setting Proxy Routes**: The **SimpleHTTPProxy.set_routes** method is called with a dictionary that maps a key to a proxy route URL. This sets up the proxy server with the necessary routing information.
- **Creating the HTTP Server**: An instance of **HTTPServer** is created, binding it to the localhost address (127.0.0.1) and port 8080. The **SimpleHTTPProxy** class is passed as the handler class, which means it will handle all incoming HTTP requests to the server.

Execution

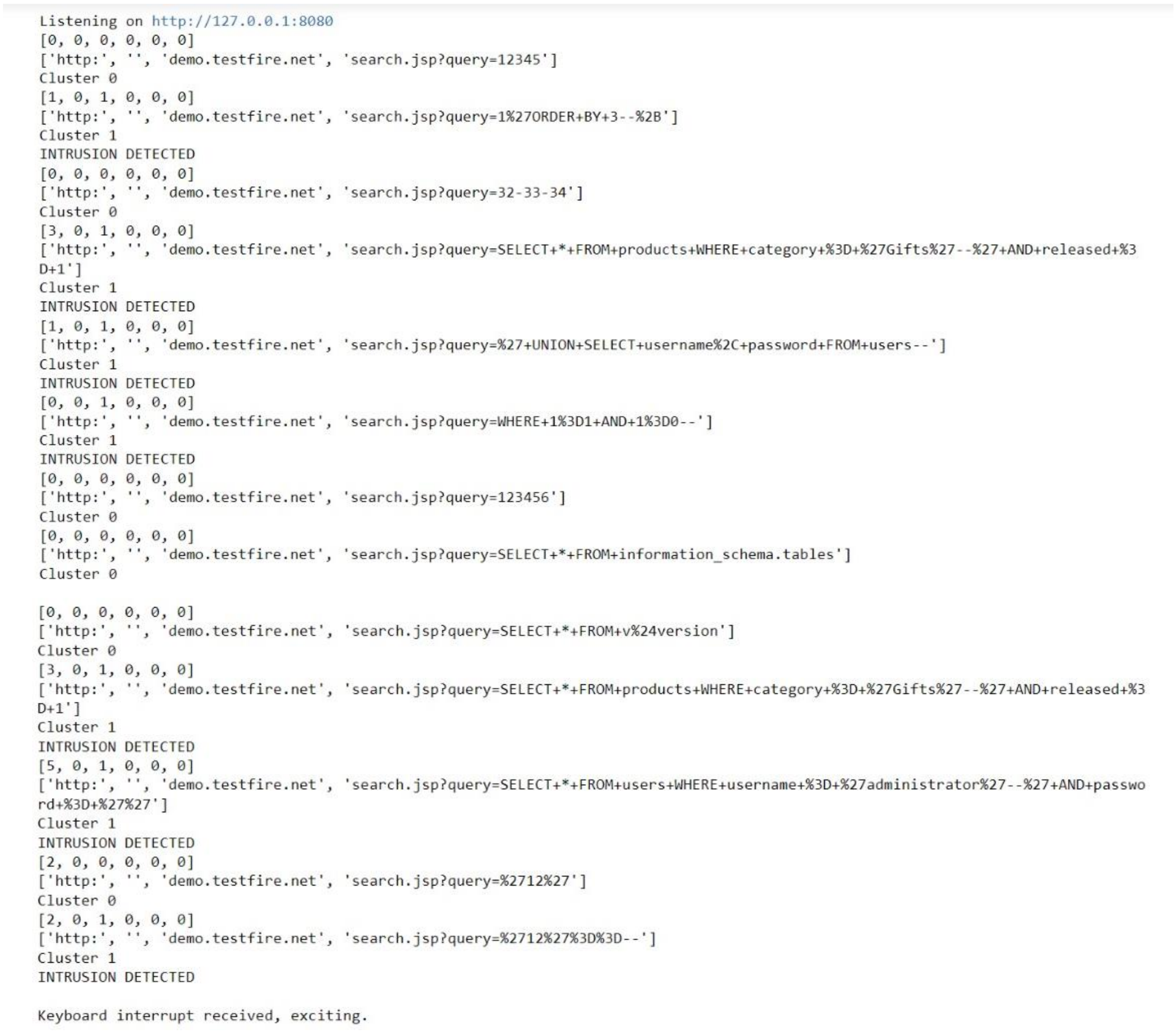
- **Starting the Server**: The server is started with the **httpd.serve_forever()** method, which begins listening for incoming HTTP requests. It runs indefinitely until it is manually stopped.
- **Listening for Requests**: The server prints a message indicating it is listening on the specified host and port, providing feedback that it is ready to receive and process requests.
- **Handling Keyboard Interrupt**: The server is designed to run continuously until a keyboard interrupt (Ctrl+C) is received. Upon receiving this signal, the server will shut down gracefully, printing a message to indicate that it has been interrupted.

Significance

- **Accessibility**: By listening on **127.0.0.1**, the server is accessible locally on the machine it is running on, which is typical for a development environment.
- **Port Configuration**: Port **8080** is commonly used for HTTP servers, especially when the default port **80** is already in use or requires elevated permissions.
- **Graceful Shutdown**: The handling of **KeyboardInterrupt** allows the server to be stopped without abruptly terminating connections, which is important for maintaining data integrity and system stability.

This process ensures that the SimpleHTTPProxy is correctly configured and capable of serving requests, providing both security through intrusion detection and functionality through request routing. It’s a crucial part of the project’s operation, enabling it to fulfill its role as a proxy server.

5.3 Output



```
Listening on http://127.0.0.1:8080
[0, 0, 0, 0, 0, 0]
['http:', '', 'demo.testfire.net', 'search.jsp?query=12345']
Cluster 0
[1, 0, 1, 0, 0, 0]
['http:', '', 'demo.testfire.net', 'search.jsp?query=1%27ORDER+BY+3--%2B']
Cluster 1
INTRUSION DETECTED
[0, 0, 0, 0, 0, 0]
['http:', '', 'demo.testfire.net', 'search.jsp?query=32-33-34']
Cluster 0
[3, 0, 1, 0, 0, 0]
['http:', '', 'demo.testfire.net', 'search.jsp?query=SELECT+*+FROM+products+WHERE+category+%3D+%27Gifts%27--%27+AND+released+%3
D+1']
Cluster 1
INTRUSION DETECTED
[1, 0, 1, 0, 0, 0]
['http:', '', 'demo.testfire.net', 'search.jsp?query=%27+UNION+SELECT+username%2C+password+FROM+users--']
Cluster 1
INTRUSION DETECTED
[0, 0, 1, 0, 0, 0]
['http:', '', 'demo.testfire.net', 'search.jsp?query=WHERE+1%3D1+AND+1%3D0--']
Cluster 1
INTRUSION DETECTED
[0, 0, 0, 0, 0, 0]
['http:', '', 'demo.testfire.net', 'search.jsp?query=123456']
Cluster 0
[0, 0, 0, 0, 0, 0]
['http:', '', 'demo.testfire.net', 'search.jsp?query=SELECT+*+FROM+information_schema.tables']
Cluster 0

[0, 0, 0, 0, 0, 0]
['http:', '', 'demo.testfire.net', 'search.jsp?query=SELECT+*+FROM+v%24version']
Cluster 0
[3, 0, 1, 0, 0, 0]
['http:', '', 'demo.testfire.net', 'search.jsp?query=SELECT+*+FROM+products+WHERE+category+%3D+%27Gifts%27--%27+AND+released+%3
D+1']
Cluster 1
INTRUSION DETECTED
[5, 0, 1, 0, 0, 0]
['http:', '', 'demo.testfire.net', 'search.jsp?query=SELECT+*+FROM+users+WHERE+username+%3D+%27administrator%27--%27+AND+passwo
rd+%3D+%27%27']
Cluster 1
INTRUSION DETECTED
[2, 0, 0, 0, 0, 0]
['http:', '', 'demo.testfire.net', 'search.jsp?query=%2712%27']
Cluster 0
[2, 0, 1, 0, 0, 0]
['http:', '', 'demo.testfire.net', 'search.jsp?query=%2712%27%3D%3D--']
Cluster 1
INTRUSION DETECTED

Keyboard interrupt received, exciting.
```

Chapter 6

Applications

1. Advanced Threat Detection:
   Detect and block SQL injection attacks in real-time, enhancing web application security.
2. Reduced False Positives:
   Machine learning minimizes false positives, ensuring accurate threat detection.
3. Adaptive Defense:
   Automatically adapt defense strategies to countering SQL injection techniques.
4. Actionable Insights:
   Provide insightful security analytics for informed decision-making and future security strategies.
5. Research and Learning:
   This projects provides an opportunity to gain hands-on experience in machine learning, cybersecurity, and network security. It can deepen their understanding of algorithms, data preprocessing techniques, model training, and deployment strategies.

Chapter 7

Conclusion

7.1 Future Scope

1. Expand detection capabilities: Enhance the project to detect and mitigate various web application attacks beyond SQL injections, including cross-site scripting (XSS), cross-siterequest forgery (CSRF), and more.

2. Continuous training: Implement mechanisms for continuous training of machine learningmodels to adapt to evolving cyber threats and improve detection accuracy over time.

3. Cloud deployment: Enable cloud deployment to provide scalability, reliability, andaccessibility, allowing users to access the project’s security features over the internet without local installation.

4. Develop a user-friendly web interface: Create an intuitive web interface to facilitate easy configuration, data uploading, monitoring of security events, and access to insights generated by machine learning models, catering to users with diverse technical backgrounds.

5. Integration with existing security systems: Explore integration opportunities with existing security systems and frameworks to complement and enhance overall cybersecurityposture for organizations and individuals.

7.2 Conclusion

In conclusion, the integration of machine learning into Web Application Firewalls (WAFs) gives a solution to enhance cybersecurity in the face of evolving threats. This study aims to improve threat detection and mitigation capabilities, building upon exist ing systems that demonstrate the effectiveness of this approach. By imposing machine learning algorithm, WAFs can better distinguish between malicious and benign activities, which strengthens defense mechanisms against cyber intrusions. Overall, this project seeks to strengthen internet security by utilizing the power of machine learning within WAFs.

Chapter 8

References

1. S. Sharma, R. Gupta, and A. Singh, ”Improving Internet Firewall Using Machine Learn- ing Techniques,” Proceedings of the International Conference on Computer Networks, Big Data and IoT, 2019.
2. R. Smith and M. Brown, ”Signature-Based and Machine Learning-Based Web Appli- cation Firewalls,” Journal of Cybersecurity and Information Assurance, Vol. 5, No. 2, 2020.
3. Y. Chen and Z. Zhang, ”A Comprehensive Survey on Machine Learning-Based Web Application Firewall Techniques,” IEEE Access, Vol. 8, 2020.
4. For operating Burp suite: https://portswigger.net/burp
5. OWASP “Open web application security project.”. Available: https://www.owasp.org
6. The Ultimate Guide to K-Means Clustering: Definition, Methods and Applications Available: https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/
7. OWASP. OWASP Zed Attack Proxy (ZAP). [Online]. Available: https://owasp.org/
8. Burpsuite Parser from https://github.com/debasishm89/burpy