DrillBit

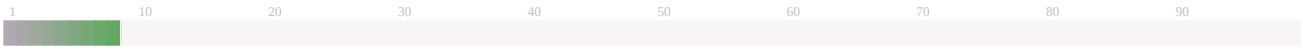
The Report is Generated by DrillBit Plagiarism Detection Software

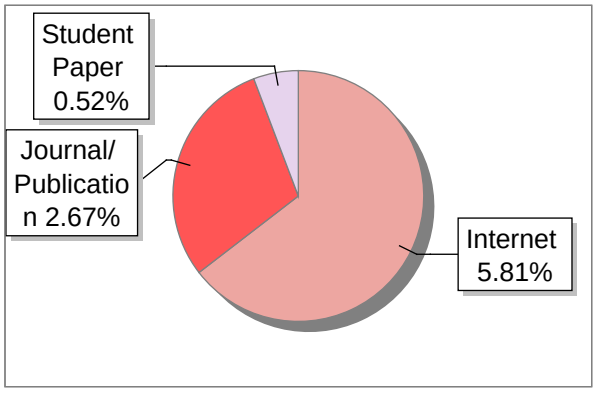
Submission Information

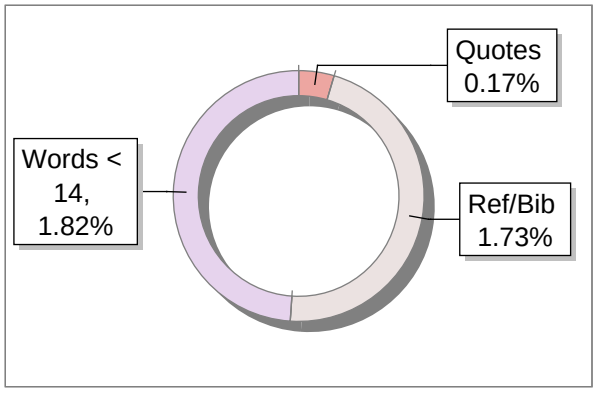
Author Name	Aalekh
Title	Web-Application Firewall using Machine Learning..
Paper/Submission ID	1097458
Submitted by	krc@iiitdwd.ac.in
Submission Date	2023-11-10 13:24:28
Total Pages	17
Document type	Project Work

Result Information

Similarity **9 %**







Exclude Information

Quotes	Excluded
References/Bibliography	Excluded
Sources: Less than 14 Words Similarity	Excluded
Excluded Source	**0 %**
Excluded Phrases	Excluded

A Unique QR Code use to View/Download/Share Pdf File

DrillBit

DrillBit Similarity Report

9	3	A
SIMILARITY %	MATCHED SOURCES	GRADE

A-Satisfactory (0-10%)
B-Upgrade (11-40%)
C-Poor (41-60%)
D-Unacceptable (61-100%)

LOCATION	MATCHED DOMAIN	%	SOURCE TYPE
1	documents.mx	3	Internet Data
2	moam.info	1	Internet Data
4	biomedeng.jmir.org	1	Internet Data

EXCLUDED PHRASES

1	**indian institute of informtion technology dharwad**

Mini Project Report on

Web-Application Firewall using Machine Learning

Submitted by

B Sri Venkata Sai Tarun (20BCS025)

Aalekh Prasad (20BEC001)

Ankit Kumar (20BCS017)

Under the guidance of

Dr. Radhika B S

Assistant Professor, Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY DHARWAD

07/11/23

Certificate

It is certified that the work contained in the project report titled "Web Application Firewall using Machine Learning" by B Sri Venkata Sai Tarun (20BCS025), Aalekh Prasad (20bec001) and Ankit Kumar (20BCS025) have been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Dr. Radhika B S

DEPARTMENT OF Computer Science Engineering

Nov, 2023

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 be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

B Sri Venkata Sai Tarun (20BCS025)

Aalekh Prasad (20BEC001)

Ankit Kumar (20BCS017)

Contents

List of Tables

1 Introduction

Web applications are essential components of modern business operations, but they are also prime targets for cyberattacks. Traditional network firewalls can filter traffic based on predefined rules but lack the agility to adapt to emerging threats. Machine learning, on the other hand, offers the potential to detect and mitigate web application attacks in real-time, making it a valuable addition to network security. However, with the proliferation of web applications, the threat landscape has evolved dramatically. Cyberattacks, particularly those targeting web applications, have become increasingly sophisticated and pervasive. Consequently, traditional network firewalls, which rely on static rule-based filtering, often struggle to keep pace with the dynamic nature of these threats.

To address this ever-growing challenge, a paradigm shift in cybersecurity strategies is required. This report delves into the innovative fusion of machine learning techniques with network firewalls to fortify web application security. Machine learning, a subset of artificial intelligence (AI), offers unparalleled potential for improving the detection, response, and prevention of web application attacks. By harnessing the power of machine learning, organizations can augment their existing cybersecurity defenses, ushering in a new era of proactive and adaptive protection.

The report will explore how machine learning, a technology that excels in pattern recognition, data analysis, and real-time decision-making, can be leveraged to bolster the capabilities of network firewalls. It will delve into the methodologies, models, and applications of machine learning in the context of network security, highlighting the advantages and challenges of this approach.

Additionally, it will provide real-world case studies showcasing the effectiveness of machine learning-powered firewalls in safeguarding critical web applications and sensitive data.

As organizations strive to secure their web applications against an ever-evolving array of threats, the integration of machine learning into network firewalls is poised to be a pivotal component of their cybersecurity strategy. This report aims to shed light on the intersection of machine learning and network security, offering insights, recommendations, and a path forward to fortify web application security in an era of growing digital threats.

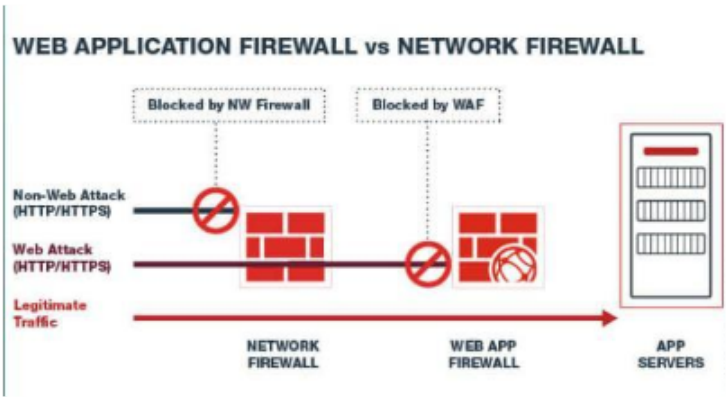


Fig (i)

2. Literature Review

2.1 Web Application Security Firewall Challenges

Web applications face an array of security challenges due to their widespread use and exposure to the internet. The literature highlights common vulnerabilities such as SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), and remote file inclusion. These vulnerabilities can lead to data breaches, unauthorized access, and other security incidents, jeopardizing the confidentiality, integrity, and availability of sensitive information.

The review also addresses the limitations of traditional security mechanisms, such as rule-based firewalls and signature-based detection systems, in adequately protecting web applications against sophisticated and evolving threats. These mechanisms often struggle to adapt to new attack patterns and lack the ability to identify zero-day vulnerabilities, making them less effective in safeguarding against modern cyber threats.

2.2 Traditional Web Application Firewalls

The literature provides insights into traditional Web Application Firewalls (WAFs) and their role in protecting web applications. WAFs act as an intermediary between users and web servers, inspecting and filtering incoming web traffic for potentially malicious content. These firewalls typically rely on predefined rules and signatures to detect and block known attack patterns.

However, traditional WAFs have inherent limitations. Their effectiveness heavily relies on accurate and up-to-date rule sets, which can become cumbersome to manage as the application's complexity and traffic increase. False positives and negatives can also be a concern, potentially disrupting legitimate traffic or failing to detect sophisticated attacks. As web application traffic diversifies and threats evolve, the need for a more intelligent and adaptive approach to web application security becomes evident.

2.3 Machine Learning in Web Application Firewall

The literature emphasizes the growing role of machine learning algorithms in bolstering web application security. Machine learning techniques have shown promise in effectively identifying and mitigating web application vulnerabilities by learning patterns and behaviors from large datasets.

Supervised machine learning algorithms, such as Support Vector Machines (SVM), Random Forest, and Deep Learning models, have been successfully applied to web application security for their ability to classify normal and malicious web traffic accurately. By analyzing historical data and learning from labeled examples, these models can adapt to changing attack patterns and detect previously unseen threats.

The review also discusses unsupervised and semi-supervised machine learning approaches, which can aid in anomaly detection and identifying zero-day vulnerabilities. Unsupervised techniques, like clustering and density-based algorithms, can help identify abnormal behavior without relying on pre-existing attack signatures.

In conclusion, the literature review highlights the critical need to address web application security challenges through more sophisticated means. Machine learning algorithms offer promising opportunities to enhance Web Application Firewalls and provide real-time, adaptive protection against an ever-evolving landscape of cyber threats. The integration of BurpSuite and machine learning techniques in this project exemplifies a novel approach to improving web application security by harnessing the potential of both traditional tools and advanced algorithms.

2.4 Network Security Challenges

Data Quality and Quantity : Machine learning models heavily rely on data. One of the fundamental challenges is obtaining high-quality labeled datasets that represent both normal and malicious traffic accurately. In many cases, labeled data for specific attack types may be limited, making it challenging to train models effectively.

Imbalanced Datasets: Imbalanced datasets, where malicious instances are far fewer than normal instances, pose a significant challenge. This can lead to models being biased towards the majority class (normal traffic) and struggling to identify rare attacks. Implementing techniques such as oversampling, undersampling, or using different evaluation metrics is essential to address this issue.

Feature Engineering : Feature engineering is critical for the success of machine learning models. Identifying the most relevant features and extracting meaningful information from raw data can be complex. Ensuring that features capture the necessary information for accurate threat detection is an ongoing challenge.

Model Overfitting: Machine learning models can be prone to overfitting, especially when dealing with complex datasets. Overfit models may perform well on training data but fail to generalize to new, unseen data. Employing techniques like cross-validation and regularization is crucial to mitigate overfitting.

Model Interpretability: The "black-box" nature of some machine learning models can be a challenge, as it may be challenging to understand why a model makes specific predictions. Interpretability tools and techniques are necessary for gaining insights into model decisions and ensuring transparency in security operations.

Real-Time Processing: Real-time processing of network traffic introduces constraints on model complexity and response time. Ensuring that models can make predictions in real time while maintaining accuracy is a demanding task.

Model Updates and Maintenance: The threat landscape is continually evolving, requiring frequent updates and maintenance of machine learning models. Implementing automated model updates and retraining pipelines is essential to adapt to new attack techniques and trends.

False Positives and Negatives: Balancing the trade-off between false positives and false negatives is a persistent challenge. An overly aggressive model may produce numerous false positives, leading to alert fatigue, while an overly conservative model may miss real threats.

Regulatory Compliance: Complying with regulatory frameworks, such as GDPR, HIPAA, or industry-specific standards, while deploying machine learning-based security solutions can be

challenging. Ensuring that data privacy and compliance requirements are met is a critical consideration.

Resource Constraints: The computational and resource requirements of machine learning models can be substantial, especially for large-scale deployments. Managing resource constraints and optimizing model performance is essential.

2.5 Machine Learning in Network security Firewall

Network firewalls play a crucial role in protecting network infrastructure from a wide range of threats, including unauthorized access, malware, and advanced persistent threats (APTs). Machine learning can enhance the capabilities of network firewalls in various ways:

1. Intrusion Detection and Prevention

Machine learning is widely used for intrusion detection and prevention systems (IDPS) in network firewalls. These systems analyze network traffic in real-time to identify both known and unknown threats. Here are some key applications:

Anomaly Detection: Machine learning models can learn the normal network behavior and identify anomalies that might indicate potential intrusions. These anomalies can include unusual traffic patterns, suspicious connection attempts, or sudden spikes in traffic.

Behavioral Analysis: ML algorithms can monitor the behavior of users and devices on the network to detect deviations from their typical activities. This is particularly useful for identifying insider threats or compromised accounts.

Signature-less Detection: Machine learning can identify new and previously unknown threats by analyzing network traffic without relying on pre-defined signatures. This is crucial for detecting zero-day vulnerabilities and APTs.

2. Threat Intelligence Integration

Machine learning can enhance network firewalls by incorporating threat intelligence data into the analysis process. This integration involves:

Threat Detection and Prioritization: ML models can process threat intelligence feeds to identify emerging threats and prioritize them based on relevance and impact.

Adaptive Rule Management: Threat intelligence data can be used to automatically update firewall rules and policies to protect against specific threats.

Predictive Threat Mitigation: By analyzing historical threat data, machine learning models can predict potential threats and adapt firewall rules accordingly.

3. Advanced Malware Detection

Machine learning can improve the detection of advanced malware and zero-day threats in network traffic:

File Analysis: ML models can analyze file attachments and downloads in network traffic to identify malicious files, even if they lack known signatures.

Sandbox Integration: Machine learning can work in tandem with sandboxing technology to identify and analyze suspicious files and links in a controlled environment.

DNS Traffic Analysis: DNS traffic can be analyzed using ML to detect signs of malware communication or data exfiltration.

4. Real-time Traffic Shaping and Prioritization

Machine learning can help network firewalls optimize traffic shaping and prioritize critical applications,

ensuring efficient network performance:
Quality of Service (QoS): ML models can analyze network traffic patterns to prioritize applications that require low latency or high bandwidth, ensuring a seamless user experience.
Load Balancing: Machine learning can help distribute network traffic evenly across multiple servers or data centers to prevent overloads and optimize resource utilization.
Traffic Monitoring and Capacity Planning: ML can provide insights into traffic trends, enabling better capacity planning and network optimization.

2.5.1. Binary Classification

Model Selection: For binary classification, our primary goal was to distinguish between intrusion and non-intrusion network connections. We employed three classification algorithms: **Logistic Regression, Random Forest, and XG Boost**. Additionally, we conducted experiments using different oversampling and undersampling techniques.

The key features-engineering techniques and model selection criteria included:
Without Over Sampling or Under Sampling on the Data:
All three models were trained without any modifications to the dataset, serving as our baseline experiment.
Using Under Sampling with NearMiss Algorithm:
Under-sampling with the NearMiss algorithm was applied to address the class imbalance in the data.
Using Over Sampling with RandomOverSampler Algorithm:
Over-sampling with the RandomOverSampler algorithm was utilized to mitigate class imbalance.
Using Over Sampling with SMOTETomek Algorithm:
Over-sampling with the SMOTETomek algorithm was employed to handle class imbalance effectively.

Model Evaluation: We evaluated the models using performance metrics, including precision, recall, F1-score, and AUC-ROC. Since the dataset was imbalanced, we placed a significant emphasis on the recall score as a critical performance metric.

Final Model Selection: After comprehensive analysis and model evaluation, the Random Forest model with oversampling using SMOTETomek was identified as the best model for binary classification. It exhibited the highest recall score, making it well-suited for identifying intrusion signals in an imbalanced dataset.

2.5.2. Multiclass Classification

Model Selection: In multiclass classification, our objective was to categorize network connections into four different types of intrusions: Probing, DOS, U2R, and R2L. We used the same three classification algorithms (Logistic Regression, Random Forest, and XG Boost) and applied various feature engineering techniques.
The feature engineering techniques and model selection criteria were consistent with those in the binary classification part.

Model Evaluation: We evaluated the multiclass models using the log loss metric and examined the misclassified points. Our aim was to minimize log loss, indicating a closer match between predicted probabilities and actual values.

Final Model Selection: Despite challenges with overfitting, the XG Boost model without oversampling or undersampling emerged as the best choice for multiclass classification based on its log loss score.

2.5.3. Final Model Integration

The final step involved integrating the best binary and multiclass models to create a unified intrusion detection system. We tested the combined model with random data to ensure it could accurately predict intrusion types and detect network anomalies.
Random Forest With Over Sampling Using SMOTETomek
XGBOOST Without Over Sampling or Under Sampling

2.5.4. Model Performance

The final models demonstrated strong performance in detecting intrusion and categorizing them into multiple classes. The models were tested against real-world network data to assess their practical usability and reliability.
The success of these models is a significant achievement in enhancing network security and mitigating cyber threats in a dynamic digital landscape.

3. Methodology

3.1 Data Collection with BurpSuite

The data collection phase involves using BurpSuite, a popular web security testing tool, to capture HTTP traffic from a demo website. BurpSuite is configured to act as a proxy server, enabling the interception of web requests and responses between users and the web server.

3.1.1 Configuring BurpSuite for Data Capture: The configuration of BurpSuite involves setting up the proxy listener, browser configurations, and SSL certificate installation (if required) to ensure seamless interception of HTTP traffic.

3.1.2 Capturing HTTP Requests and Responses: As users interact with the demo website, BurpSuite captures and logs the HTTP requests and responses, storing them in a log file for further analysis and processing.

Fig (ii)

3.2 Data Preprocessing

Data preprocessing is crucial to prepare the captured log files for machine learning model training. The preprocessing steps involve cleaning and structuring the data for subsequent feature extraction.

3.2.1 Log File Parsing: The raw log files obtained from BurpSuite are parsed to extract relevant information such as request URLs, headers, parameters, and response codes. This step aims to organize the data into a structured format suitable for subsequent analysis.

3.2.2 Extracting HTTP Parameters: From the parsed data, HTTP parameters are extracted from each request to identify the input data for the machine learning models. This process involves identifying parameters in query strings, form data, and cookies.

3.3 Data Labeling and Dataset Creation

In this phase, the extracted HTTP parameters are labeled based on different vulnerability types to create the labeled dataset used for model training and evaluation.

3.3.1 Selection of Vulnerability Types: A predefined set of common web application vulnerabilities, including SQL injection, XSS, CSRF, and others, is selected for labeling the dataset. The choice of vulnerability types is based on their significance and relevance to the project's objectives.

3.3.2 Labeling Requests for Different Vulnerabilities: Each HTTP parameter is assigned a label indicating the vulnerability type it represents. Parameters that exhibit characteristics of a particular vulnerability are labeled accordingly, while benign parameters are labeled as non-vulnerable.

3.4 Feature Engineering

Feature engineering involves transforming the labeled HTTP parameters into a suitable feature representation that can be used as input for the machine learning models.

3.4.1 Parameter Encoding Techniques: The extracted parameters are encoded into numerical representations using techniques such as one-hot encoding, tokenization, or hashing. This process converts the parameters into feature vectors, enabling their use in machine learning algorithms.

3.4.2 Feature Selection: Relevant features are selected to train the machine learning models. Feature selection techniques, such as correlation analysis or information gain, are employed to retain the most informative features and eliminate redundant ones.

4. System Design and Architecture

4.1 Overview of Web Application Firewall

The Web Application Firewall is designed to enhance web application security by leveraging machine learning algorithms to detect and mitigate potential threats in real-time. The WAF acts as an intermediary between users and the web server, inspecting and analyzing incoming HTTP requests to identify malicious activities.

The main components of the Web Application Firewall include:

Data Collection Module: The data collection module is implemented using BurpSuite, which captures HTTP traffic from a demo website. It intercepts requests and responses, logging them for further analysis.

Data Preprocessing Module: This module parses the captured log files, extracting HTTP parameters and organizing the data into a structured format suitable for feature engineering and model training.

Machine Learning Model Module: Machine learning algorithms are employed to analyze the extracted HTTP parameters and classify them as either benign or malicious. The system utilizes a selection of supervised and unsupervised learning techniques to achieve accurate threat detection.

Decision Engine: The decision engine is responsible for determining the action to be taken based on the model's classification results. It decides whether to block or allow incoming requests based on their predicted vulnerability types.

Response Mechanism: The response mechanism communicates with the web server to block or allow requests based on the decision made by the decision engine. This ensures real-time threat detection and immediate response to potential security threats.

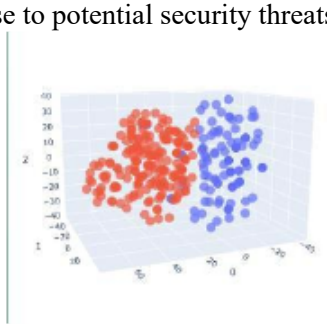


Fig (iii)

4.2 Python Script for Data Extraction

To enable seamless data extraction from BurpSuite, a Python script is developed as an integral part of the system. The script performs the following tasks:

Configuring BurpSuite Proxy Listener: The script configures BurpSuite as a proxy listener to intercept and log the HTTP traffic between users and the web server.

Parsing Log Files: Once the log files are captured by BurpSuite, the Python script parses these log files to extract relevant information such as HTTP requests, responses, URLs, headers, and parameters.

Extracting HTTP Parameters: From the parsed data, the script identifies and extracts HTTP parameters from each request. These parameters are crucial for training the machine learning models.

The Python script acts as a bridge between BurpSuite and the subsequent phases of the project,

facilitating the smooth flow of data from data collection to preprocessing and feature engineering stages.

5. Data Analysis and Preprocessing

5.1 Exploratory Data Analysis

5.1.1 Overview of Classification

Train Data: The training dataset contains 4,898,431 data points with 42 features.
Test Data: The test dataset consists of 311,029 data points with the same 42 features.
Memory Reduction: To manage the high memory usage, a memory reduction function was applied, significantly reducing memory usage.

Overall, data cleaning and preprocessing tasks included null value removal and duplicate value elimination.

5.1.2 Binary Classification

In the binary classification part: Class Label Modification: The class_label was modified to classlabel_bi for binary classification.
Imbalance Observation: Both datasets exhibit significant class imbalance, with non-intrusion signals dominating.

In the training dataset, non-intrusion signals constitute approximately 81.898 percent , while intrusion signals make up the remaining 18.102 present.

In the test dataset, non-intrusion signals account for approximately 71.083 percent, and intrusion signals are around 28.917 percent.

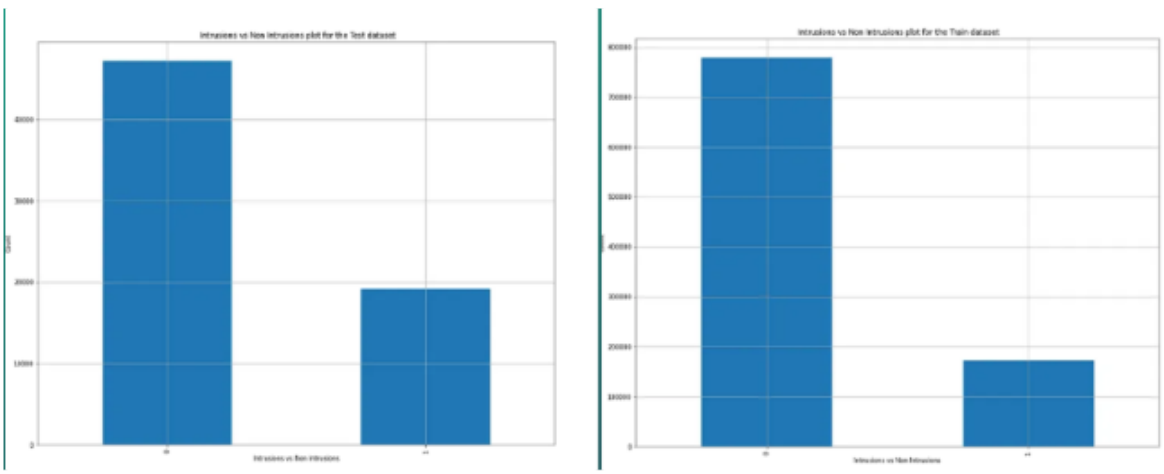


Fig (iv)

5.5.3 Multiclass Classification

For the multiclass classification part:Class Label Modification: The 'class_label' was modified to 'classlabel_mul' for multiclass classification.
Imbalance Observation: The analysis indicated that DOS type intrusion is prevalent in both the train and test datasets. Specific features, such as protocol_type, service, and flag, demonstrated variations in the type of intrusions they attract.

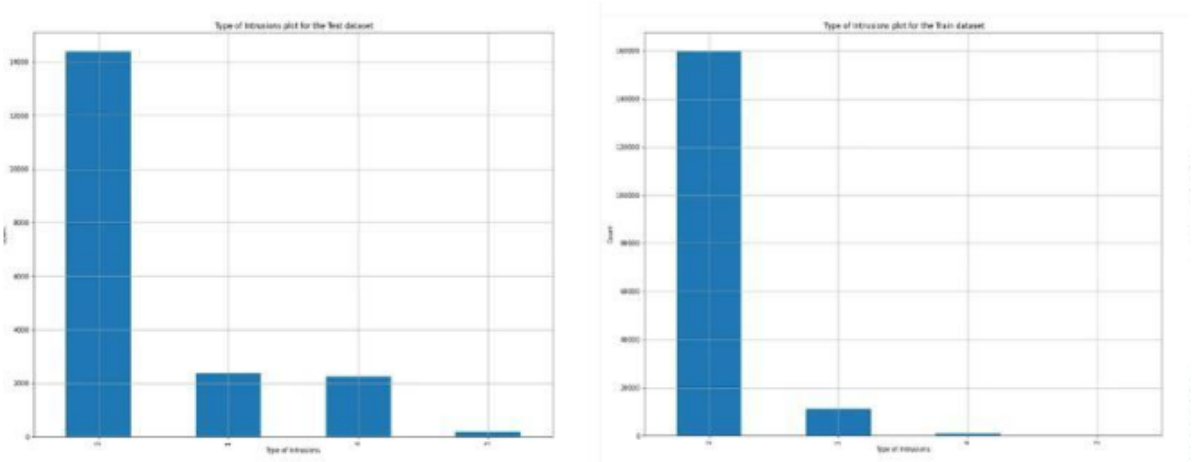


Fig (v)

5.1.4 Analysis of Categorical Features

Categorical features, including 'protocol_type,' 'service,' and 'flag,' were analyzed for both binary and multiclass classification. Key observations include:
'protocol_type': The presence of three different types of protocols (TCP, UDP, ICMP) was noted in both datasets. TCP protocol had a higher number of non-intrusion signals compared to intrusion signals, whereas ICMP protocol had slightly more intrusion signals.
'service': The majority of services occurred very frequently, with the top services covering 90 percent of the training data. Specific services attracted different types of intrusions.
'flag': In both datasets, specific flags, such as SF, REJ, S0, and RSTO, were more associated with intrusion responses.

5.1.5 Analysis of Continuous Features

Several continuous features were analyzed using visualization techniques, such as PCA, t-SNE, and clustering. Key observations include:
'duration': Most data points were concentrated around 0, and intrusion signals had occasional spikes.
'srcbytes': Both class labels had challenging-to-analyze patterns, with some intrusion signals exhibiting high data transfer spikes.
'dstbytes': The test dataset showed several large spikes in non-intrusion signals (class label 0).
'wrongfragment': Data points mostly did not fluctuate, with only a few outliers present in class label 1.
The application of PCA, tSNE, and clustering techniques revealed insights into data separability and overlap for both binary and multiclass classification.

6. Results and Conclusion

The integration of machine learning in web application firewalls (WAFs) and network firewalls represents a significant advancement in the field of cybersecurity. This report has explored the development, deployment, and real-time monitoring of machine learning applications to enhance security measures. As we conclude, it's imperative to underscore the key takeaways from this study.

6.1. Effective Threat Mitigation

Machine learning, with its ability to analyze vast amounts of data in real time, offers an effective approach to identifying and mitigating cybersecurity threats. By continuously learning and adapting, ML-based WAFs and network firewalls can swiftly detect and respond to evolving attack techniques.

6.2. Improved Accuracy

The use of machine learning can significantly enhance the accuracy of threat detection. ML models can analyze complex patterns and anomalies, reducing false positives and negatives. This translates to a more precise identification of threats and a reduction in alert fatigue for security teams.

6.3. Real-Time Monitoring

Real-time monitoring of network traffic is essential in today's dynamic threat landscape. Machine learning algorithms are well-suited for this task, providing rapid threat detection and response capabilities. This is crucial for preemptively addressing security incidents as they occur.

6.4. Challenges and Considerations

While machine learning holds great promise for enhancing cybersecurity, there are challenges to address. These include the need for high-quality data, handling imbalanced datasets, feature engineering, model interpretability, and the ongoing maintenance of models. Organizations must carefully navigate these challenges to harness the full potential of ML-based security solutions.

6.5. Regulatory Compliance

Compliance with data privacy regulations and industry standards remains a critical consideration in the deployment of machine learning-based security solutions. Organizations must ensure that data handling and model training processes adhere to relevant legal and regulatory frameworks.

6.6. Continuous Improvement

In the realm of cybersecurity, complacency is not an option. As threats evolve, so must security measures. The continual refinement and updating of machine learning models, coupled with real-time monitoring and adaptive responses, are essential for staying ahead of adversaries.

7. Future Scope

I. **Deep Learning in Web & Network Security:**
   - Deep learning, a subset of Machine Learning, holds immense promise in web and network security.
   - Deep neural networks are capable of understanding complex patterns and are being harnessed to detect even more sophisticated threats.

II. **AI-Driven Web Application Firewalls (WAFs):**
   - The integration of Artificial Intelligence (AI) with WAFs is on the horizon.

III. **Cross-Platform Support:**
   - Extend machine learning-based security solutions to support various operating systems, devices, and platforms to provide comprehensive protection.

IV. **Evasion Techniques Research:**
   - Continually research and develop evasion techniques that can help identify and

Bibliography

1. https://www.appliedaicourse.com/
2. https://arxiv.org/pdf/1903.02460.pdf
3. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
4. https://tryhackme.com/room/introwebapplicationsecurity
5. https://www.sciencedirect.com/topics/computer-science/network-based-intrusion-detection-system
6. https://owasp.org/www-community/
7. M. Małowidzki, P. Berezinski, M. Mazur, Network Intrusion Detection: Half a Kingdom for a Good Dataset, in: NATO STO SAS-139 Workshop, Portugal, 2015.
8. W. Haider, J. Hu, J. Slay, B. Turnbull, Y. Xie, Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling.
9. Journal of Network and Computer Applications 87 (2017) 185–192.doi:10.1016/j.jnca.2017.03.018 _Application_Firewall