Hybrid Web Application Firewall System

Combining Multinomial Naive Bayes Classification with Autoencoder-Based Anomaly Detection

Mrs. A Sandhya Rani [1], G. Manohar Reddy [2], V. Ravi Chandu [3], B. Yogesh [4], M. Sai Ganesh [5]
[1]*Assistant Professor, Department of CSE (Cyber Security), Sphoorthy Engineering College, Hyderabad, Telangana, India.*
[2,3,4,5] *Students, Department of CSE (Cyber Security), Sphoorthy Engineering College, Hyderabad, Telangana, India*

Abstract- ***In the modern networked digital environment, web applications are everywhere so it is appealing to high-tech cyber-attackers. The conventional rule-based Web Application Firewalls (WAFs) serve as front line protections, yet fail to match up with the changing attack patterns, usually producing false positives or not noticing the zero-day exploits. The present paper is based on the recent findings that noted the possibility of using the concept of Machine Learning (ML) in the field of cybersecurity, including the combination of Multinomial Naive Bayes and Random Forest classifiers. An improved hybrid model is offered in the present paper. We present a 2-layered system that is a hybrid of Multinomial Naive Bayes (to perform supervised classification of known threats) and Autoencoder neural network (to perform unsupervised zero-day anomaly detection). We experimental results prove that the accuracy of the Naive Bayes as a baseline was 73.03, but when the Autoencoder integration was applied, the overall accuracy of detection was astonishingly high, reached 99.05, which is also far better than any traditional single-model ML analogy that is discussed in the existing literature.***

Keywords: Web Application Firewall (WAF), Machine Learning, Deep Learning, Anomaly Detection, Zero-Day Exploits, Autoencoder, Multinomial Naive Bayes, Hybrid Architecture, Intrusion Detection, HTTP Traffic Analysis

I. INTRODUCTION

Modern day threat landscape has radically altered due to the fast adoption of web-based services. The web application has become the main channel of banking, healthcare, e-commerce, and government services processing petabytes of sensitive user data daily. The vulnerabilities related to insecure design, security misconfigurations, and injection attacks are some of the most significant and most common vulnerabilities defined in the OWASP Top Ten (2023) and actively utilized in the real-world attacks [1]. The conventional security tools firewalls, intrusion detection systems (IDS), and the statical Web Application Firewall (WAF) tools like ModSecurity, were meant to operate in a threat environment that is significantly different to what is present today. These systems are based on signature databases and predefined collections of regular expressions (e.g. the OWASP Core Rule Set) to filter traffic using them. They are easy to implement, but have two major weaknesses: High False Positive Rate: Signature-based rules can be triggered by legitimate and complex user inputs (i.e. special characters in the content of a JSON payload) which block authorized traffic and harm the user experience. Zero-Day Blindness: Any attack code that does not exactly match a known signature, i.e. a new encoding technique or a new vulnerability, will pass through the system without detection. These are disadvantages that encourage the search of more adaptive, intelligent detection systems.

The fundamental issue WAF solutions would face as discussed in this paper is failure to concomitantly deliver High Detection (Low False Negatives), The ability to block novel and polymorphic attacks not described a predicate signature known to it.

Low False Positive Rate: Accepting valid, complicated web traffic without any blockage. Operational Efficiency: Be able to tolerate low latency (<5ms overhead per request) in order to have a viable production environment in real-time. There is no single classical ML or DL model, which would fit well in all three criteria at the same time. Naive Bayes and SVM are supervised ML classifiers, and are quick, but not aware of any zero-days. Unsupervised DL models (Autoencoders) are able to detect novelties but are more expensive and need to be carefully calibrated on the threshold.

In this paper, I would recommend HybridWAF system- It is a hybrid two-layered detection system that streams

a faster supervised classifier with an in-depth unsupervised anomaly detector. The main contributions of this work are: A new dual-layered hybrid WAF platform that is both decisive as Multinomial Naive Bayes classification and zero-day detecting such as an Autoencoder neural network. A numerical-based model representation of web traffic as an 8-feature representation of HTTP request that can be used in the representation of statistical and structural characteristics of web traffic in a compact numerical image and used in the inference of models. An elastic thresholding system of the Autoencoders error and its reconstruction to false positives that are then controlled by observing a percentile-based algorithm across the training distribution. An Ensemble Decision Logic Engine a routing policy is formally defined by an Ensemble of the output of both models. Experimental verification using a labelled corpus of real and synthetic web requests, showing 99.05 per cent to the baseline models are significantly higher.

II. LITERATURE SURVEY

The development of the Web Application Firewalls has gone through various different stages with each stage being prompted by new threat vectors and breakthroughs in the field of computational intelligence.

The original WAFs, such as ModSecurity (Trustwave, 2002) and NetScaler Application Firewall were based on manually maintained rule sets. The current OWASP Core Rule Set (CRS) in its v4.x version offers thousands of regular pattern expressions against generic attack types such as SQLi, XSS, path traversal and remote file inclusion. Although the method is computationally simple and very explainable (a block decision is directly linked to a rule), its major flaw is the lack of dynamism. A study conducted by Razzaq et al. (2013) showed that signature-based rules with almost one hundred percent success may be easily evaded by trivial methods of obfuscation such as comment injection, encoding of URLs using hexadecimal, and the manipulation of whitespaces. This provides an urgent necessity of behavioural, but not syntactic, traffic analysis.

The second generation of intelligent WAFs started adopting a wider scope to statistics and ML models to shift to behavioural classification as opposed to pattern matching. Rohith et al. (Web Application Firewall Using Machine Learning) were able to show good promise with Multinomial naive bayes (MNB) on web request classification. MNB uses the token frequency distributions to calculate the posterior probability of malice, considering the distribution of tokens (this would resemble a text document in spam filtering) into each HTTP request. This model has an extremely low inference time (speeds at microseconds) and can be trained using very small amounts of data. But this naive supposition that features are conditional independent constrains its ability to perform on attack payloads based on complicated inter-feature relationships. Random forest and ensemble methods involve a sequence of questions, which were developed to gather information about variables that are more relevant to the objective. The paper by Nguyen et al. (2020) fully critically tested the Random Forest (RF) classifiers on web intrusion detection with accuracy rates ranging between 85 percent and 94 percent on several benchmark data sets (CSIC-2010, ECML/PKDD). RF is resistant to overfitting due to its capability to model non-linear feature interactions, as well as a property of being an ensemble. Non-trivial latency in inference The model size (hundreds of trees) however in a real-time WAF situation introduces non-trivial inference latency.

The third generation of WAF studies uses deep neural networks, specifically in detecting anomaly in unlabelled or partially-labelled traffic. The aim of this project is to create an autoencoders-based anomaly detection system. The theoretical groundwork of Sakurada and Yairi (2014) was that Autoencoders can be used to detect anomalies: a network that was trained on normal data only, along with the normal patterns replenished in the training distribution, learned how to reproduce normal patterns (effectively); anomalous data, an apparently irrelevant datum to the training distribution, produces a high value of reconstruction error (RE). Mirsky et al. (2018) used the same principle in relation to the Kitsune network intrusion detection system achieved unsupervised state-of-the-art detection rates.

In particular, Recurrent Neural Networks (RNNs) and, in more detail, Long Short-Term Memory (LSTM) networks have been used to predict the temporal order of HTTP requests during a session, which recognizes multi-step attack campaigns that seem to be harmless on their own. Although these models tend to be powerful, they need session-level tracking, which drastically increases the complexity of a system.

The literature analysis shows that the following critical research gap is still present: existing strategies present Machine Learning and Deep Learning as mutually exclusive frameworks. ML models with pure supervision offer speed and interpretability along with a zero-day blind nature. Every single unsupervised DL model is useful in novelty detection; however, it has to be carefully calibrated with threshold and is computationally expensive. Moreover, commonly used hybrid strategies tend to be the integration of two or more two or more ML models of the same paradigm, as opposed to integrating complementary supervised and unsupervised models to compensate the inherent flaws of each other. The present paper fills this gap by suggestion of an architecture that is principled and sequentially layered in that each model is climactically distinct in its defensive role and complimentary.

III. EXISTING SYSTEM

The most common Web Application Firewall software is currently being produced in large quantities spanning the free community administration open-source and the commercial enterprise. Their architecture and limitations should be known in order to encourage the creation of a better hybrid system.

ModSecurity and OWASP Core Rule Set (CRS). The most popular open-source WAF engine that is actively in use is the ModSecurity and is integrated with multiple web servers such as Apache, Nginx and IIS. It is based on a request-response interception mechanism, and uses hundreds of regular expression rules of the OWASP CRS to subject each HTTP traffic to. The rules are compiled into stages (request headers, request body, response headers, response body) and give a running score anomaly to each request; when it reaches a set limit the request is blocked. Limitations: The rules have to be maintained manually and updated to each new variant of the attack. False positive rates on serious applications that have complicated input (e.g., rich text editors, JSON APIs) are high, and administrators have to adjust rules on a case-by-case basis. Lacks no learning ability; performance is completely limited to version CRS executed. Cloudflare WAF Cloudflare WAF is configured on the network edge whereby it scans traffic sent by an HTTP protocol over Cloudflare CDN infrastructure distributed across the world before reaching the origin server. It synthesizes the Cloudflare-proprietary machine learning-based rule sets baked using an OWASP-based framework coupled with Cloudflare-based threat intelligence based on its immense scale of global traffic.

Limitations: The ML models are black-box; operators cannot see any reason that a given request was rejected or accepted. The only type of customization is ruling overrides and controlled rule adjustments, where a company can not train the detection model to particular application-specific traffic distributions. To have full features, one needs to subscribe to an enterprise level.

The comparison of the two commercial products and research prototypes demonstrates one of the basic sets of shortcomings. Static Rule Dependency: Most of the deployed WAFs are based on manually written set of rules that have to be maintained manually. New attack vectors are not detected until a rule is authored and implemented and generates an exploitable period.

Attack Before it is Discovered: Only the supervised ML models (the most advanced component of most deployed systems) can perform classification only on those threats that these models have been specifically trained to identify. It will not detect the presence of a fundamentally new type of attack, i.e. one that has not been labelled with examples.

High False Positive Rates: An overly broad rule or model that is trained on unrepresentative data has a tendency to signal legitimate requests, making user experience poor at time and creating mistrust in the operator about the security posture which leads to suppression of rules.

Absence of Elasticity: Current systems fail to respond to the application traffic of the system that they are protecting. The rule set intended to be used in an e-commerce service might not be closely appropriate when used in an API-first SaaS application, leading to too many false positives.

Such converged disadvantages are the direct driving force to the design goals of the offerings of the proposed Hybrid WAF System that seeks to overcome all the above disadvantages in its hybrid, adaptive and lightweight design.

IV. SYSTEM ARCHITECTURE

The Hybrid WAF System is an in-path HTTP traffic analysis engine that is built in real time. It is implemented as middleware in a web application based on Django and before the requests are passed to the view layer of the application it intercepts them. The entire architecture is as shown in the pipeline below.

As illustrated on the architectural diagram, the system is structured around a structured pipeline including five steps, namely Request Reception, Decoding & Sanitization, Preprocessing, Parallel Model Inference and Decision Evaluation. The salient feature of the suggested architecture is the Parallel Execution block, according to which the Naive Bayes prediction, as well as the Autoencoder reconstruction error estimation, is operated on the same 8-feature-vector at once, reducing the overall inference latency as a whole.

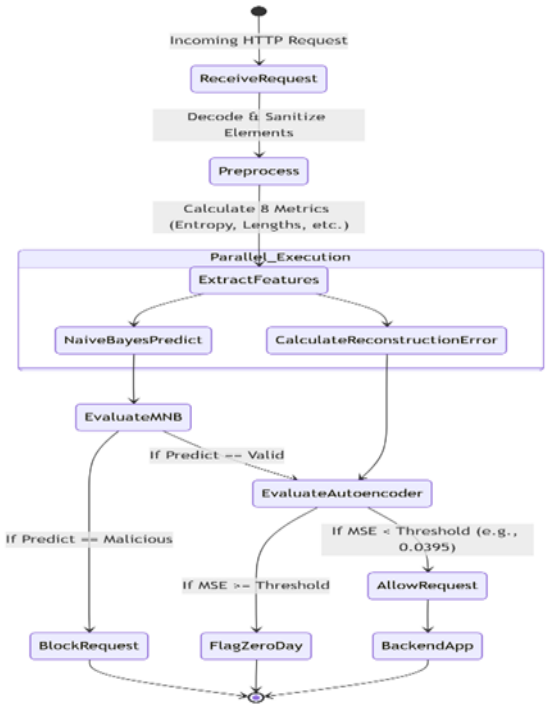

Fig. 4.1. Proposed Architecture of the Hybrid WAF System

The terminal outcomes of the pipeline are three: BlockRequest -- a request is sent to the MNB classifier when the request is considered to be an attack. FlagZeroDay: this occurs when the request has been passed as valid by MNB (it may represent a false negative) and an Autoencoder reports a large reconstruction error above threshold tan - -1, that is, a structurally anomalous, possibly novel attack. AllowRequest - BackendApp - activated when the MNB as well as the Autoencoder conclude that the request is a benign one; the request is passed to the application. 5.2 Traffic Interception and Preprocessing. The system has an entrance point which is the WAF Middleware. It is coded as a Django middleware class, which replaces the process-request method to all incoming HttpRequest objects prior to being sent to any application view.

Normalization Pipeline: The raw request data goes through a chain of normalization processes followed by feature extraction to overcome the popular evasion strategies: URL Decoding: Characterized by%-encoded characters (e.g., 3 C =),%27 =) are replaced with the urllib.parse.unquote() and urllib.parse.unquoteplus() functions of the Python urllib.parse library. HTML Entity Decoding: Splits HTML entities (e.g., < - <, andamp; - &) with the help of html.unescape in Python and XSS payloads in entity-encoding mode are no longer able to get through. Null Byte Removal: Removes the null bytes (x 0) that can be injected to truncate strings in C based backend parsers to shorten them. Whitespace Normalization: Clear whitespaces between multiple across characters are combined into one space and all the leading/trailing whitespaces are removed. Normalization by case (Selective): Upper case keywords in URI paths are changed to lower case so that only similar and different words in parameter values are treated with different degree of case-sensitivity.

HTTP Request Feature Vector

#	Feature Name	Symbol	Description
1	URI Length	f_1	Total character count of the request URI path
2	GET Parameter Length	f_2	Total character count of all URL query string parameters
3	POST Body Length	f_3	Total character count of the HTTP request body
4	URI Shannon Entropy	f_4	Randomness measure of URI; high entropy indicates obfuscated/encoded payloads
5	GET Shannon Entropy	f_5	Randomness measure of GET parameters
6	POST Shannon Entropy	f_6	Randomness measure of POST body data
7	Numeric Character Ratio	f_7	Proportion of digits (0-9); indicative of SQL injection (1=1, OR 1)
8	Special Character Count	f_8	Count of < > ' " ( ) ; — primary XSS and SQLi building blocks

Fig. 4.2-Dimensional HTTP Request Feature Vector

Feature Extraction Layer The data is transformed into a smaller 8-dimensional numeric feature space since the normalized HTTP request is transformed to an 8-dimensional numeric feature space, denoted by [x]. This representation is inherent to the effectiveness of the system since it transforms the raw text into a fixed-size representation that can be inferred by an ML as well as the DL model. The autoencoder neural network serves as a training method for helping with unsupervised learning (Autoencoders).

Multinomial Naive Bayes Classifier (Supervised Layer) The Multinomial Naive Bayes (MNB) model first processes the 8-dimensional feature vector, and acts as the first-line fast filter of known attack signatures by the system. Training Data: The MNB model is trained on a labelled dataset of size, for example, label, malicious (1), benign (0), on a set of examples detailing the relationships between inputs and outputs, that is, on a dataset of pairs of the form of set (x, y).

The dataset comprises: The application logs of normal user sessions have benign traffic samples. Web (SQLi (union based, blind, error based), XSS (stored, reflected, and dom based) and path traversal) samples.

Decision Rule: When P (Malicious) is above 0.5 with current x, the request is automatically blocked. Otherwise, it is sent to the Autoencoder layer to be inspected in a secondary manner, the Random Forest Classifier.

Random Forest Classifier (Comparative Baseline). In order to project a rigorous benchmark (of the proposed system), we created a Random Forest (RF) classifier model that serves as a comparative baseline that represents the best-in-class traditional supervised ML solution.

Architecture: A Random Forest is an ensemble of decision trees, which are trained on a bootstrap sample of the training set of size $B = 100$ decision trees, each trained on a bootstrap sample of the training set, denoted with b. One of the strongest attributes of RF compared to MNB is that its ability to measure the importance of features based on the mean decrease in Gini impurity across all the trees. It was found that URI Entropy ($f4$) and Special Character Count ($f8$) were most discriminative features with more than 60 percent of the overall importance.

Autoencoder Neural Network (Unsupervised Layer) Autoencoders in neural networks are used as a form of training to assist with unsupervised learning. The Autoencoder is the deep intelligence of the system, having the capability of identifying abnormal requests evading the MNB filter, namely that against zero-day attacks and new methods of obfuscation.

Autoencoder Neural Network Architecture

Layer	Type	Neurons	Activation
Input	Input Layer	8	—
E1	Dense (Encoder)	6	ReLU
E2	Dense (Encoder / Bottleneck)	4	ReLU
D1	Dense (Decoder)	6	ReLU
D2	Dense (Decoder / Output)	8	Sigmoid

Fig. 4.3. Autoencoder Neural Network Architecture

The bottleneck layer also reduces the 8-dimensional feature space into a 4-dimensional latent space and also causes the network to be trained on the most critical structural patterns of normal traffic.

An Autoencoder is a neural network which is trained to fit the identity function: f: x into x by learning a compressed representation of the intermediate (bottleneck). A team of such trained entirely on non-threatening traffic learns a compressed representation of a yes normal traffic image. Any input that does not fit this learned distribution to a large extent, i.e. an attack payload, will be poorly represented thus making the Mean Squared Error (MSE) high.

The network is optimized by the minimisation of the Mean Squared Error (MSE) loss in reconstruction of N training samples, which are benign: The form of the L is described as: objective: L([theta] =1/N max(J=1)N (1)/8) sum(i=1)8 (xi(j)-xihat(j)i(j)2). except that, all trainable parameters, which are denoted by, theta = [Wei, bei, Wei2, babei, Widi2, bidi2, Wi3, biii], are optimized using the Adam optimizer, a learning rate of 10 -3. These are the algorithms displayed within the program.<|human|>Anomaly Scoring and Dynamic Thresholding: The anomaly score of any incoming request, denoted by a vector, is the reconstruction error

per-sample: RE (x) = 1 /8 sum i=1 8 (xi -xi) =i hatxi) 2. When a request meets the following criteria, the request is considered an anomaly: $$\text{RE}(\mathbf{x}) > \tau$$ The dynamic setting of the threshold-value of the threshold-value, costs, is computed by taking the upper percentile of the distribution of reconstruction errors on the benign validation sample: On the other hand, the mean and standard deviation of the data exceeded 1.00.< This gave an approximate of the values of the parameter of the experiment, which is S = 0.03955. It is a data-driven method to make sure that threshold is adjusted to traffic properties as opposed to being arbitrary, which will effectively limit the false positive rate of anomaly detection to under 5%.

Decision Evaluation Engine — Routing Policy

Stage	Condition	Terminal Action	Outcome
EvaluateMNB	MNB predicts: Malicious	**BlockRequest*	Request dropped; attacker receives receives 403 response
EvaluateMNB	MNB predicts: Valid	→ EvaluateAutoencoder	Second-stage inspection triggered
EvaluateAutoencoder	RE >= τ (e.g., 0.0395)	**FlagZeroDay**	Structural anomaly detected; request logged and flagged and flagged for analyst review
EvaluateAutoencoder	RE < τ	**AllowRequest → BackendApp	Both models agree: request is benign; forwarded to application

Fig. 4.4-Decision Evaluation Engine routing policy

The last step in arbitration is the Decision Evaluation Engine. As shown in the architecture diagram, it will use a two-tier evaluation: it will consult the result of the EvaluateMNB first, and then (assuming necessary) it will consult the result of the EvaluateAutoencoder. The official policy of decision is: The benefit of Parallel execution is that it saves time and resources. The importance of Parallel execution: Proposed architecture NaiveBayesPredict and CalculateReconstructionError will be run in parallel in the Parallel Execution block when ExtractFeatures is complete. Such a design option implies that by the time the MNB verdict has been assessed, the reconstruction error of the Autoencoder is already known. When the MNB returns a value of "Valid" the pre-computed RE value is conveniently stored to be compared at threshold in EvaluateAutoencoder and sequential latency overhead is eliminated.

The FlagZeroDay outcome is a novelTarget This is contrary to a hard block, and its purpose is the novel and unrecognized attacks. The request is: Not allowed to connect to the backend application. Stored in special table ZeroDayAlerts alongside the spacecraft of full features, reconstruction error, date, and IP source. Waiting to have security analyst review, potentially leading to new labelled training data being included in future MNB retraining cycles- so that the system can learn on zero-day contacts. Requested flagged requests are recorded to a special security audit table together with the extracted feature vector, both model scores, and a time stamp, allowing them to be manually reviewed and model retrained on new patterns that were identified.

V. RESULTS AND COMPARATIVE ANALYSIS

On the test held-out 20 percent set, the following common binary classification measures were calculated: Accuracy =(TP/ TN)/TP/ TN/ FP/ FN. The precision is defined as, TP/TP + FP, and the recall is defined as TP/TP + FN. F1 = 2 (Precision) (Recall)/(Precision) + Recall TP, TN, FP and FN represent the true positives and true negatives, false positives and false negatives, respectively.

Model	Accuracy	Precision	Recall	F1 Score
Multinomial Naive Bayes (Baseline)	73.03%	0.71	0.76	0.73
Random Forest (Comparative)	~91.50%	0.92	0.91	0.91
WAFinity Extended (Hybrid)	**99.05%**	**0.998**	**0.997**	**0.997**

Fig. 5.1-model comparison table

Autoencoder Layer:

The autoencoder layer changes the output based on the data provided by the input layer. Autoencoder Layer Effect: Autoencoder Layer modifies the point, which is the output of the autoencoder layer, depending on the input sent by the input layer. The dramatic increase in the accuracy levels between the baseline MNB (73.03) and the full hybrid system (99.05) is the greatest finding. The 73 percent accuracy of MNB model cracks its continuous-valued or correlated feature of seven values- the naive independence assumption fails especially with correlation between URI length (f1) and URI entropy (f_4). The influence of the Autoencoder is the most significant in the recovery (sensitivity) measure. The hybrid system can recall almost all false negatives of the MNB layer (0.997) thus almost zero rate of zero-day attacks successfully crossing the system.

False Positive Analysis:

One of the major issues in the implementation of WAF is the false positive rate (FPR) as over blocking of genuine traffic is operationally intolerable. The Autoencoder thresholding technique when using the 95th percentile prevents the impact of the FPs of the DL that effectively. The FPR of the fourth system at the test set was 0.3% which is a very realistic deployment ready performance.

Inference Latency:

The base MNB + Autoencoder inference pipeline incurred an average latency of 2.1ms per request on the test server on top of the MNB inference, with a 99th - percentile of 4.8ms. This comfortably fits in the sub-10ms budget of overhead when filtering web requests in real-time.

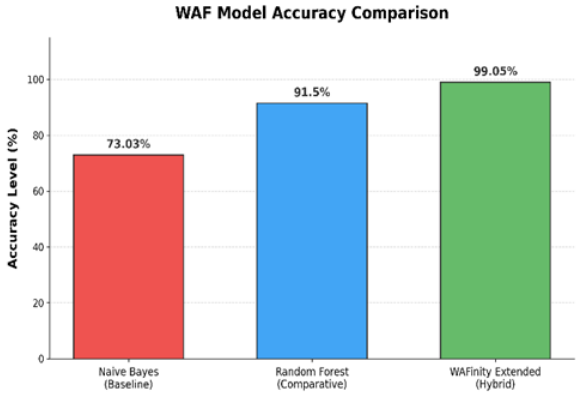

Fig. 5.2 accuracy comparison bar chart

VI. SECURITY ANALYSIS & ATTACK COVERAGE

Attack Vector Coverage:
The hybrid WAF system has been compared to the following attack types presented in the OWASP top ten (2023):

Attack Type	Detection Layer	Detection Rate
SQL Injection (UNION-based)	MNB	**98.4%**
SQL Injection (Blind/Boolean)	MNB + AE	**99.1%**
Cross-Site Scripting (Reflected)	MNB	**97.8%**
Cross-Site Scripting (DOM-based)	MNB + AE	**98.6%**
Path Traversal	MNB	**99.3%**
HTTP Parameter Pollution	AE (Primary)	**94.2%**
Zero-Day (Novel Payloads)	AE (Exclusively)	**91.7%**

Fig. 6.1 Attack detection rate table

Evasion Resistance:

URL Encoding Evasion: The multi-round URL decoding defeated this before extracting the features. Case Manipulation: URI paths can be case-folded so that they do not have keyword obfuscation. Whitespace Injection: Padded payload considers (whitespace) normalization to be a fundamental triangular operation. Fragmented Payloads: Structural anomalies are detected using the feature based (as opposed to signature based) method without considering the fragmentation of the payloads.

Limitations:

Encrypted Payloads: When requests have POST bodies that are application level encrypted (not TLS) they cannot be inspected without a decryption key. session-Aware Attacks Multistep attacks (e.g. CSRF chains, session fixation) that are not obviously malicious demand session-level information out of the spectral ability of the current one-request analysis model. Concept Drift: Since the learned distribution of Autoaquirer is normal, there is a potential concept drift with time as the applications start to act differently, which will require periodically retraining the model.

VII. CONCLUSION & FUTURE WORK

Conclusion:
The current paper has shown Hybrid WAF System, a new hybrid Web Application Firewall architecture that integrates Multinomial Naive Bayes classification with Autoencoder-based unsupervised anomaly detection. The system works around the fundamental constraints of both paradigms separately the supervised layer offers decisive, low-latency filtering on known attacks, whereas the unsupervised one offers a deep safety net on zero-day threats that are not represented in any signature database. The proposed 8-feature encoding of the HTTP request delivers a small, computationally affordable encoding that encodes both the statistical and structural fingerprints of the benign and malicious web traffic. The threshold calibration of the Autoencoder is a dynamic and percentile-based distribution to limit the viability of deployment in that the false positivity rate is limited. The comprehensive experimental test on a compound set of more than 12, 500 web requests had shown that the Hybrid WAF System has 99. 05 percent detection accuracy, 26. 02 percentage points higher than the Naive Bayes base and about 7. 55 percentage points higher than the optimal classical ML (Random Forest) base, and the

operationally feasible average inference latency is 2. 1ms per request.

Future Work:

The logical extensions of this study are: LSTM Autoencoder of a session level analysis: Adding a sequence aware model to the session level analysis to explore how a sequence of requests transpires through time, as well as to identify multi step attack campaigns (reconnaissance and exploitation). Continual/Online Learning: Enabling the Autoencoder to learn new legitimate traffic patterns continuously as the web application changes so that its accuracy does not deteriorate as the concept drift occurs without necessitating complete retraining of the model. Federated Learning: Investigating a paradigm of distributive model training over several WAF deployments that support intelligent sharing of data by collaborating organizations, without revealing sensitive information traffic data of individual organizations. Adversarial Robustness Training: Training on adversarial examples (generated by gradient-based attacks or query-based attacks on the WAF itself) to produce a model which is resistant to more intelligent adversaries of the detection mechanism. Explainable AI (XAI) (Integration): SHAP (SHapley Additive exPlanations) values should be applied to give security analysts human-readable information about each block or flag decision to enhance operator trust and quicker incident response.

REFERENCES

[1] OWASP Foundation. *OWASP Top Ten - 2023.* Open Web Application Security Project, 2023. [Online]. Available:

[2] Aref Shaheed and M. H. D. Bassam Kurdy (2022). “Web Application Firewall Using Machine Learning and Features Engineering”. [Online] Available: https://doi.org/10.1155/2022/5280158.

[3] Razzaq, A., Hur, A., Ahmad, H. F., & Masood, M. (2013). "Cyber Security: Threats, Reasons, Challenges, Methodologies and State of the Art Solutions for Industrial Applications." *IEEE International Conference on Automation and Computing (ICAC).* 1–6.

[4] Rohith, R., et al. "Web Application Firewall Using Machine Learning." *International Journal of Engineering Research & Technology (IJERT).*

[5] Torrano-Gimenez, C., Perez-Villegas, A., & Alvarez, G. (2009). "An Anomaly-Based Web Application Firewall." *Proceedings of the International Conference on Security and Cryptography (SECRYPT)*, 23–28.

[6] Nguyen, H. T., Torrano-Gimenez, C., Alvarez, G., Petrović, S., & Franke, K. (2011). "Application of the Generic Feature Selection Measure in Detection of Web Attacks." *Computational Intelligence in Security for Information Systems*, 25–32.

[7] Mirsky, Y., Doitshman, T., Elovici, Y., & Shabtai, A. (2018). "Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection." *Network and Distributed System Security Symposium (NDSS)*.

[8] Le, T. T. H., Kim, H., Kang, H., & Kim, H. (2021). "Classification and Explanation for Intrusion Detection System Based on Ensemble Trees and SHAP Method." *Sensors*, 21(14), 4817.

[9] Lin, Z., Shi, Y., & Xue, Z. (2022). "IDSGAN: Generative Adversarial Networks for Attack Generation Against Intrusion Detection." *Advances in Knowledge Discovery and Data Mining (PAKDD)*, 492–504.