IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

A Machine Learning–Driven Honeypot Framework For Real-Time Web Application Intrusion Detection

Degala Upanisha [1], Yatham Srinivas [2], Gopu Srihari [3], Nammi Hita Srikara [4], Mr. S. Nagendra [5]

[1] B.Tech-CSE Student, [2] B.Tech-CSE Student, [3] B.Tech-CSE Student, [4] B.Tech-CSE Student,

[5] Assistant Professor, M.Tech(Ph.D)

[1,2,3,4,5] Department of Computer Science and Engineering,

[1,2,3,4,5] Aditya College of Engineering and Technology, Surampalem, Andhra Pradesh, India.

Abstract: Web applications are vulnerable to malicious acts because the applications are open and interactive. SQL injection, cross site scripting, command execution, and directory browsing are among the most frequent and destructive attacks that are injection based. Traditional security mechanisms, though, tend to be very based on hard-coded signatures and firewall rules, do not provide any form of contextual observation and in-depth behavioral analysis. In the current paper, a framework of a Machine Learning-based honeypot in combination with session-based audit logging is introduced to provide real-time web application security monitoring. The suggested architecture is an implementation of a threat score mechanism that uses controlled honeypot endpoints to receive potentially suspicious inputs and rate incoming requests based on those. Besides payload analysis, the structure logs session identifiers, IP addresses, geolocation information, and repeated patterns of interaction to give more understanding of the way an attacker operates. All the identified events are registered in a long-lasting audit database and presented in the form of a responsive analytics dashboard. The practical confirmation of the provided framework by simulated attack cases proves the fact that it is a successful approach to detecting threats related to injections without notable deployment costs.

Index Terms - Web Honeypot, Supervised Learning, Attack Classification, Risk Scoring, Intrusion Detection, Web Application Security, Large Language Models, Threat Intelligence

I. INTRODUCTION

To facilitate the services of e-commerce, banking, education, and healthcare, web applications are also a highly important part of digital infrastructure and are used extensively. With the further increase in the use of online platforms, the exposure of these systems to cybersecurity disabilities increases. The most common and most dangerous ones are injection-based attacks that are SQL injection, cross-site scripting (XSS), command injection, directory traversal.

Web applications are usually secured using traditional security systems like firewalls and signature detection systems of intrusions. Nonetheless, these systems are largely based on set guidelines and familiar patterns of attacks. Although they are efficient in terms of dealing with threats that have been identified in the past, they might not be well-furnished to track and study the behavioral pattern of attackers. In most instances, suspicious requests are just blocked without any detailed logging or contextual analysis and hence the administrators cannot learn about strain of attackers or detecting recurrent intrusion attempts.

Honeypot systems have seen the light to create increased visibility of web-based threats in order to proactively monitor these threats. A honeypot is a monitored setting that creates a simulation of vulnerable endpoints and encourages malicious traffic. Honeypots can offer important knowledge about real-life attacks by capturing attack payloads and other metadata and not revealing the actual system components. In contrast

to the conventional prevention-based approaches, this strategy is based on observation, analysis and a regulated monitoring.

In addition to payload analysis, session-level behavior tracking, and contextual data (IP addresses, geographical source, etc.) are important aspects that can supplement the threat analysis. The recurrent attacks with the same location or session may signify the attempts of targeted intrusion. This monitoring strategy when used together with systematic audit logs will allow the security administrators to analyze the pattern, frequency, and distribution of attacks more efficiently.

This paper suggests a ML-based honeypot architecture with session-aware audit logging, which allows web application security monitoring in real-time. The system uses administrated honeypot endpoints to accept and process suspicious inputs through an ordered system of threat scoring. Collected events are added to a persistent database and viewed using an analytics dashboard to offer actionable security information. The suggested framework will provide a simple, scalable, and work-friendly system of enhancing web application security monitoring.

II. EXISTING SYSTEM VS PROPOSED SYSTEM

Existing System

The most common type of traditional web security implements is the use of the static rule based firewall systems, signature-based intrusion detection systems (IDS) and manual log analysis. Its methodologies only identify familiar attacks and have difficulty with dynamic or obfuscated code. Raw log generation within most systems lacks meaningful analytics and thus real-time threat intelligence is challenging. Moreover, the risk evaluation is usually binary (attack or no attack) and is not supported by contextual scoring or covert behavior. This inhibits preemptive protection and minimizes the capacity of detecting new trends of attacks.

Proposed System

The suggested system also proposes the implementation of an intelligent honeypot-based framework of monitoring with a machine-based classification of the payloads and the dynamic risk-based scoring. It does not use fixed signatures instead it works with incoming payloads based on trained models to identify patterns of malicious behavior. The system models the session, trend analytics of attacks and threat severity-based scoring to deliver actionable intelligence. Attack distributions, trends and high-risk indicators are also visualized in real time using interactive dashboards. Such hybrid methodology can improve the detection accuracy, increase the visibility as well as proactive security decision-making.

III. RELATED WORK

The use of honeypots and intrusion detection mechanisms has also received a lot of research and has been found to be effective in tracking and identifying cyber threats. The alternate concept introduced by Provos and Holz was virtual honeypots to track the activity of botnet and study the behavior of attackers in controlled settings. Their study revealed that honeypots are capable of gathering intelligence information regarding any imminent threats without subjecting actual production systems to danger. Nevertheless, due to their nature of data gathering, most of the existing implementations of a honeypot do not have an embedded system of monitoring controls or an organized system of threat scoring and rarely do they have a mechanism to efficiently analyze the information obtained.

Martin Roesch had come up with Snort which was a lightweight signature-based intrusion detection system which identifies malicious traffic based on predefined rules and pattern matching techniques. Although signature-based systems such as Snort are very efficient in identifying known attacks, they need constant updates in order to be relevant due to the introduction of new threats. They might also have problems with spotting altered or obfuscated payloads.

Guidelines on intrusion detection and prevention system (IDPS), including systematic monitoring, detailed logging, and alert generation on time, were later given by Scarfone and Mell. In their work, they emphasized the importance of organised logging and thorough analysis of security events in order to defend it. Web application security research has also suggested rule-based input validation and filtering as a way of warding off injection attacks. They can be used to enhance immediate blocking capabilities, but since they cannot capture the context of a session or replicate an intrusion attempt made by the same source they do not allow administrators to learn about the behavior of an attacker in the long term.

Regardless of these developments, there is a relative lack of literature on the combination of honeypot-based data collection, structured rule-based analysis, session-based behavior tracking, geolocation enrichment, cumulative threat scoring and visualization of real time analytics in a single lightweight framework. The proposed system is based on the current technologies and is aimed at the structured monitoring and contextual analysis which provides the practical solution applicable in the conditions of modern web applications.

IV. METHODOLOGY

4.1 Proposed Architecture

The architecture proposed is based on the hybrid security intelligence model, which combines several elements such as web-based honeypot, session modelling engine, machine learning classification module, risk scoring and real-time analytics dashboard. The system is to operate in five interconnected layers such as the data collection layer that is controlled by the honeypot controller, the session modelling layer, the machine learning classification engine, the risk scoring and severity analysis module, and the analytics and visualization dashboard. These layers interact with each other to guarantee effective flow of data and analysis interactions. All the system components are communicated with the help of RESTful APIs, which can be described as modular, scalable, and flexible and enable each module in the system to perform independently but to be closely related to the rest of the system as well.

4.2 Web Honeypot Design

The web honeypot is priority executed as an imitated login server that replicates the action of a weak web application so as to attract potential attackers. The main aim of it is to intercept malicious interactions without subjecting the actual resources of the system to danger. Each time a user communicates with the honeypot, the system logs detailed request data including the method of the HTTP request, contents of that request, X-Forwarded-For IP addresses, information about the user-agent, dates and times, and distinct identifiers of a session. Instead of upon access, the honeypot records every interaction attempt in a safe manner, where they can be further examined and analyzed. This method allows one to gather actual world malicious payloads, in the process keeping the integrity of the systems intact and unaltered.

4.3 Session Modelling

In order to match individual requests with meaningful sessions, session modelling correlates individual requests into meaningful sessions by assigning each incoming request with a unique session identifier. This process enables the system to monitor the behavior of attackers across a number of interactions rather than viewing individual incidences. The session model derives valuable contextual data that includes geolocation of IP address, Autonomous System Number (ASN), length of session, state of the session and risk score aggregation. With these features, the system is able to create a more comprehensive view of the user or attacker behaviour and this enhances the accuracy and efficiency of the analysis that follows.

4.4 Classification and Risk Analysis

Once session modelling is done, the honeypot intercepts each request and deciphers the payload part in the request. The input payload is preprocessed and transformed into the feature representations of TF-IDF vectorization into numerical form. These characteristics are then presented to a trained multi-class machine learning classifier which can make a prediction of the probable type of attack and give the probability scores of each prediction. On the basis of these output, a dynamic risk score and a severity level is calculated to reflect the threat that the request may cause. When the prediction confidence of the classifier is less than the preset threshold, an auxiliary analysis module is launched to conduct a further analysis. The ultimate outputs are recorded and constantly monitored in the observation dashboard, which can track the threats, visualize and create warnings in real time.

4.5 Algorithm:

Algorithm 1: Intelligent Payload Classification and Risk Scoring

Procedure DETECT_ATTACK(payload P)

1. Load trained ML classifier M and TF-IDF vectorizer V
2. Preprocess input payload P
3. Transform payload into feature vector
   $VP \leftarrow V.transform(P)$
4. Predict attack class
   $C \leftarrow M.predict(VP)$
5. Compute class probabilities
   $Prob \leftarrow M.predict\_proba(VP)$
6. Determine confidence score
   $Conf \leftarrow max(Prob)$
7. Find benign class probability
   $B \leftarrow Prob(benign)$
8. Compute base risk score
   $R\_base \leftarrow (1 - B) \times 100$
9. Assign severity weight based on class C
   $W \leftarrow SeverityWeight(C)$
10. Compute weighted risk score
    $R \leftarrow min(100, R\_base \times W)$
11. If Conf < Threshold then
    Call Algorithm 2 (LLM Adjustment)
    Update class C and risk score R
12. Store prediction result in session log
13. Return class C, confidence Conf, risk score R

End Procedure

Algorithm 2: LLM-Assisted Severity Adjustment

Procedure LLM_ADJUSTMENT (payload P, risk R)

1. Send payload P to LLM analysis module
2. Extract semantic attack interpretation
3. Estimate contextual severity level
   $S \leftarrow LLM\_Severity(P)$
4. Map severity level into weight
   $W\_llm \leftarrow SeverityWeight(S)$
5. Adjust risk score
   $R \leftarrow min(100, R \times W\_llm)$
6. Return updated risk score R

End Procedure

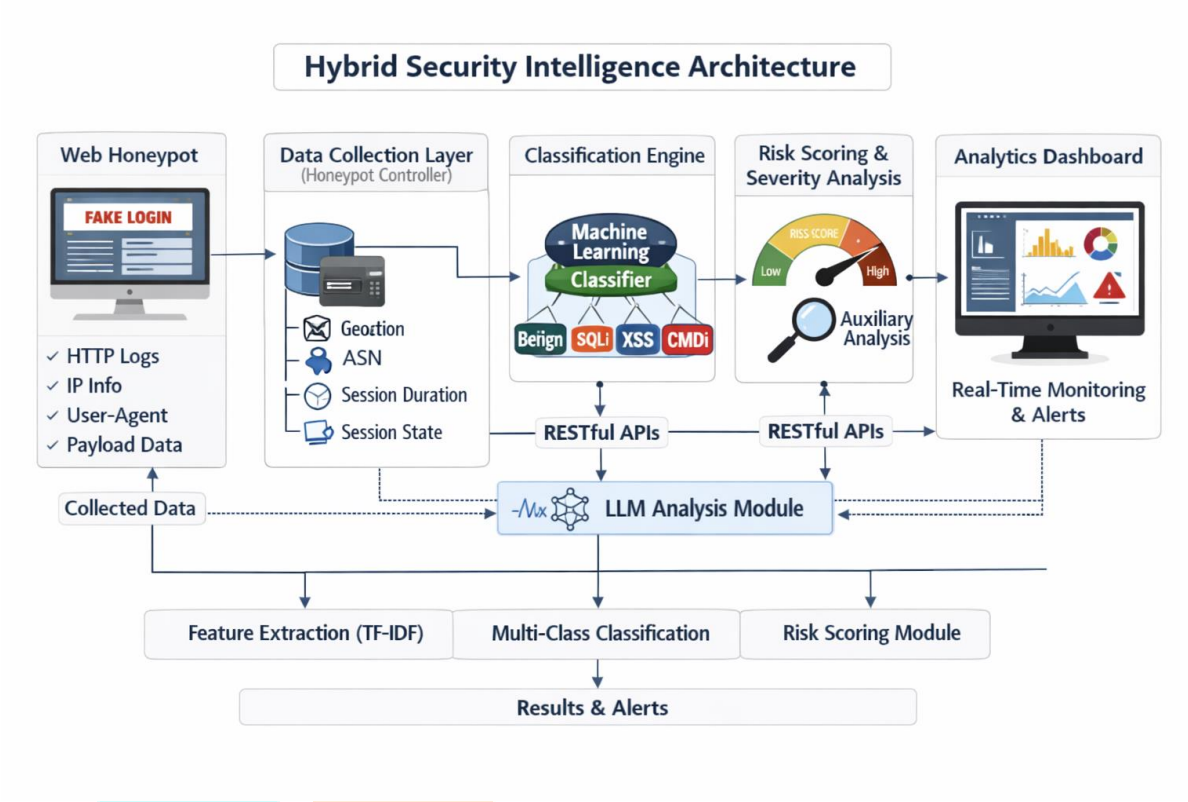

Fig 4.1 Proposed ML-Based Honeypot Intrusion Detection Architecture

V. RESULTS AND DISCUSSION

5.1 Data Preparation and Preprocessing

An individualized training set of web payloads was designed and streamlined to succeed in the classification model training. The data contained clean payload formatting as well as several malicious injection parameters namely SQL injection, cross-site scripting, command injection, brute force, and path traversal attack. The robustness was enhanced by including noisy and obfuscated variation of payloads as well, and the model was able to address the evasion techniques in the real world.

In preprocessing, the HTTP protocol metadata was pared down to get only the raw payload content that is to be analyzed. The obtained payloads were standardized by lowercasing and standardizing the text and duplicates were eliminated to avoid bias. The effort was made to ensure that classes were balanced to prevent skewed learning. Such measures guaranteed that the performance of the model was based on realistic classification ability as opposed to forced increase in accuracy due to skewed or duplicate data.

5.2 Feature Extraction

TF-IDF vectorization technique was applied to the textual payload data to convert the data into numerical representations. This started with the process of tokenization wherein the payload strings were decomposed into significant tokens. Term frequency scores were obtained to determine the significance of words in single payloads and an inverse document frequency scale was used to decrease the value of high-frequency words that were not particularly informative. The resulting feature image was simplified to a small number of numerical expressing an image that could be used in machine learning algorithms. This method retained the importance of critical attack key words and well managed the payloads of different length and structure.

5.3 Experimental Setup

The suggested system was deployed locally in a test environment. The backend was written in Java Spring Boot and the machine learning engine was written in Python through the Scikit-learn library. The feature vectorization technique was TF-IDF and a multi-class supervised model was trained to classify the payloads into various types of attacks. The frontend was a real-time monitoring and visualization security dashboard built on React. The machine learning engine was implemented as a REST API service and linked with the backend to classify the payloads on the fly and calculate the risk scores accordingly.

5.4 Data description and evaluation measures

The data were provided in the form of labeled web payloads divided into benign traffic, SQL injection, cross-site scripting, command injection, brute force, and path traversal attacks. To evaluate the model, the data was split into 20 and 80 percent training and test data respectively, which allowed carrying out a reliable evaluation of the modeling generalisation ability.

The standard metrics of model performance such as accuracy, precision, recall and F1-score were used to assess model performance. Along with these measures of classification, the probabilities of the model were also employed to justify the risk scoring system so that performance measurement was not limited to mere categorical prediction.

5.5 Results of Classification and Risk Scoring Mechanism

The trained multi-class classification model attained an overall accuracy of about 85 percent which shows equal sensitivity over major attack categories. Model performed better on obfuscated and variant payloads, which means that it can generalize well in various situations. The findings proved the separation of benign and malicious inputs successfully and with consistent multi-class discrimination.

The risk scoring was calculated on the basis of the following formula:

$$\text{Risk} = (1 - P(\text{benign})) \times 100$$

It is a probability-based approach, which produces dynamic values of risk which are proportional to the probability of malicious behavior. The multiplication of severity was done according to the type of attack to reflect actual impact levels in the real world. The value of risk created by SQL injection and command injection attacks was higher because of possible severity of the attack whereas the path traversal and cross-site scripting generated a moderate level of risk. Benign inputs had always yielded low risk scores, which illustrate the fact that the system is dynamically adjusted to the attack probability as well as attack strength.

5.6 Hybrid Classification Strategy

A hybrid classification mechanism was adopted to increase the detection strength. In case the prediction confidence of the main machine learning classifier is below a predetermined threshold, the payload is sent to a Large Language Model (LLM) to be analyzed in context. The secondary layer offers more semantic information especially when dealing with complex or ambiguous inputs. The hybrid strategy increases the resistance to the previously unknown attack patterns, massively obfuscated payloads, and semantically manipulated inputs. The statistical machine learning with contextual intelligence gives the system a greater adaptability and resilience in dynamic threat environments.

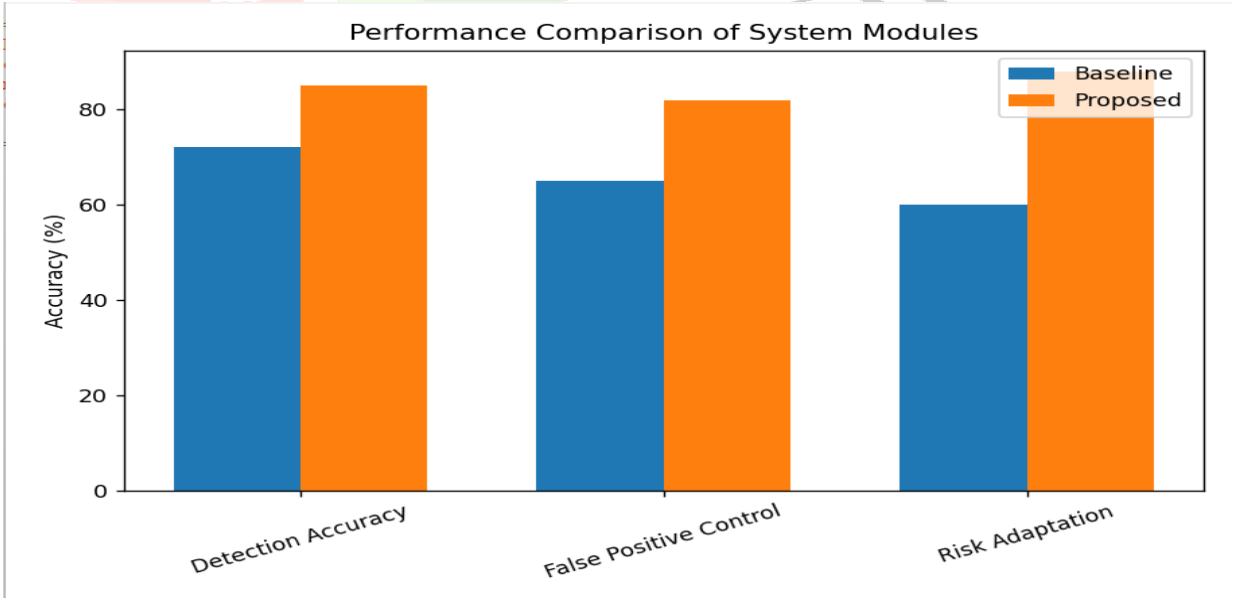

Fig 5.1 Performance Comparison of System Modules of the Baseline and Proposed System

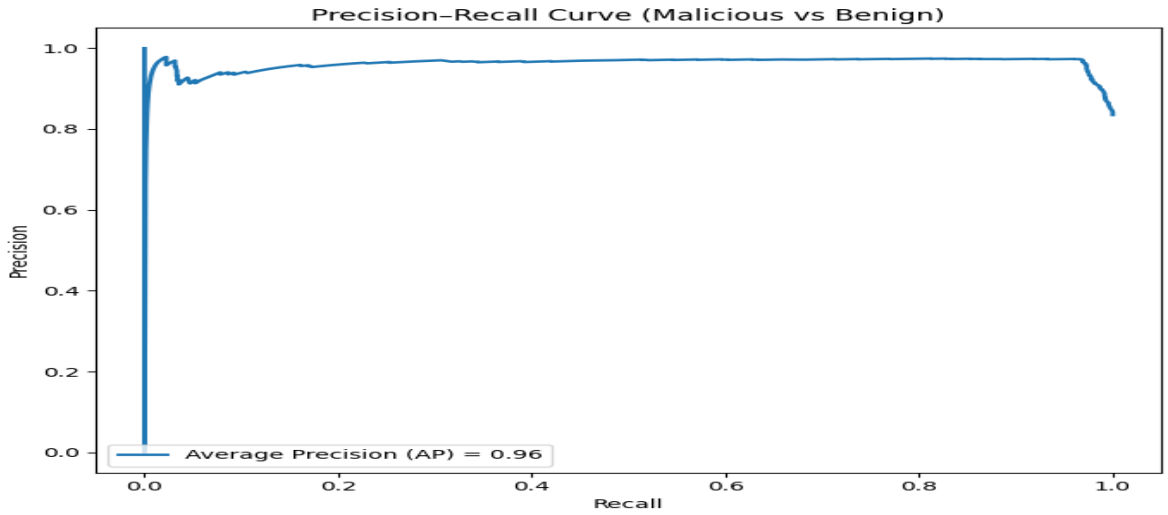

Fig 5.2 Precision – Recall curve for Malicious vs Benign Requests in the Proposed System

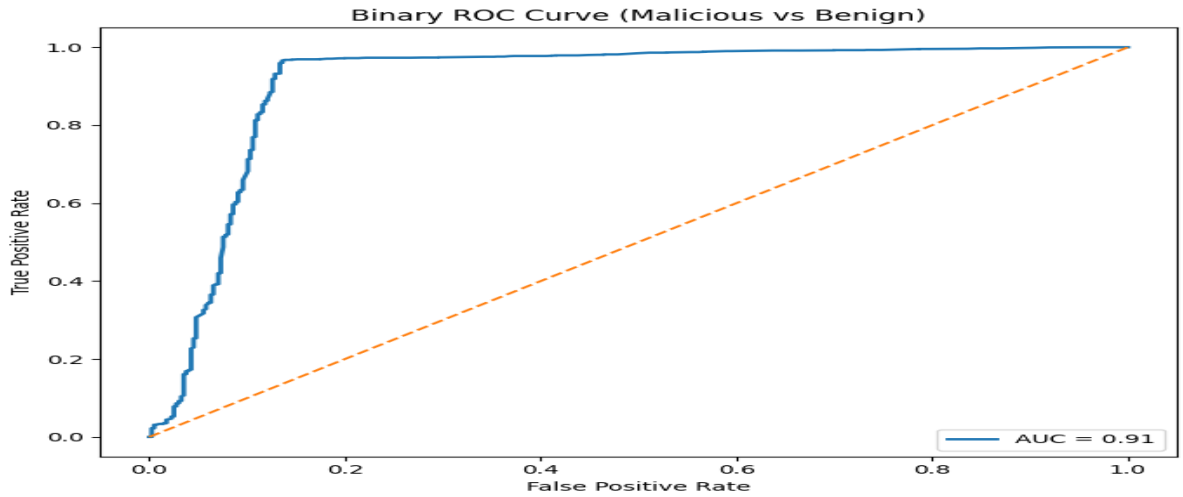

Fig 5.3 Binary ROC Curve for Malicious vs Benign Payloads in the Proposed System

5.7 Response Time Analysis:

The average time taken to process a query in a medium-sized document collection was less than three seconds. Even as the number of documents increased, the response time showed only minor variation. This indicates that the system scales well and remains responsive under increased workload. Such performance confirms that the system is suitable for real-time interaction and practical deployment.

5.8 Hybrid Retrieval Strategy:

To further improve robustness, a hybrid retrieval strategy was adopted. When the similarity confidence between a user query and stored embeddings falls below a predefined threshold, a Large Language Model is invoked for deeper contextual analysis.This hybrid approach improves the system's ability to handle ambiguous queries, complex multi-sentence questions, previously unseen document structures, and conversational inputs. By combining semantic embeddings with generative models, the system produces responses that are more accurate, meaningful, and human-like.

VI. COMPARISON WITH EXISTING SYSTEMS

Feature	Traditional WAF	Signature-Based IDS	ML-Based Detector	Proposed System
Static Rule Detection	✓	✓	✗	✓
Zero-Day Attack Detection	✗	✗	✓	✓
Payload-Level Classification	Limited	Limited	✓	✓
Multi-Class Attack Detection	✗	✗	✓	✓
Risk Scoring Mechanism	✗	✗	Limited	✓
Severity-Based Threat Level	✗	✗	✗	✓
Real-Time API Deployment	Limited	Limited	✓	✓
Honeypot Integration	✗	✗	✗	✓
Adaptive Learning Capability	✗	✗	✓	✓
LLM-Assisted Analysis	✗	✗	✗	✓
Alert Prioritization	Limited	Limited	Limited	✓

VII. FUTURE SCOPE

The suggested system can be further developed by adding massive capacity to deploy real-time as models can be continuously retrained with the recently received payload data to adapt to changes in the cyber threats. Further enhancements can be made by adopting more sophisticated deep learning systems to gain a stronger contextual insight into attack patterns, using explainable AI systems to enhance interpretability, and extending the severity-based risk scoring system into a general threat intelligence system. Furthermore, the connection to SIEM systems to enable automatic detection and response to the incident and exploration of federated learning tricks to train in a privacy-preserving fashion can greatly enhance the scalability, flexibility, and practical applicability of the system.

VIII. CONCLUSION

This paper is about a smart honeypot-based web monitoring security system, which integrates real-time attacks logs, session tracking and machine learning-based payload classification to identify and evaluate malicious activities. The system offers proactive threat intelligence to take action by merging automated risk scoring with attack type distribution analysis and live security analytics dashboards. This is because the hybrid method of statistical modeling and adaptive classification not only increases the detection but also ensures efficiency in the operation. In general, the suggested framework shows a viable and scalable system to enhance the security of web applications with the help of intelligent attack detection and monitoring based on analytics.

IX. REFERENCES

[1] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Macia-Fernandez, and E. Vazquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Computers & Security*, vol. 28, no. 1–2, pp. 18–28, 2009.
[2] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 dataset," in *Proc. IEEE Symp. Computational Intelligence for Security and Defense Applications (CISDA)*, 2009.
[3] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. Int. Conf. Information Systems Security and Privacy (ICISSP)*, 2018.
[4] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Learning to detect malicious URLs," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, 2011.
[5] A. Patcha and J. M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Computer Networks*, vol. 51, no. 12, pp. 3448–3470, 2007.
[6] OWASP Foundation, "OWASP Top 10: The Ten Most Critical Web Application Security Risks," 2021. [Online]. Available: https://owasp.org/www-project-top-ten/
[7] S. Axelsson, "The base-rate fallacy and its implications for intrusion detection," *ACM Transactions on Information and System Security*, vol. 3, no. 3, pp. 186–205, 2000.
[8] K. Scarfone and P. Mell, "Guide to intrusion detection and prevention systems (IDPS)," *NIST Special Publication 800-94*, 2007.
[9] W. Lee and S. J. Stolfo, "Data mining approaches for intrusion detection," in *Proc. USENIX Security Symposium*, 1998.
[10] C. Kruegel and G. Vigna, "Anomaly detection of web-based attacks," in *Proc. ACM Conf. Computer and Communications Security (CCS)*, 2003.
[11] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.
[12] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Proc. IEEE Symp. Security and Privacy*, 2010.
[13] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems," in *Military Communications and Information Systems Conference (MilCIS)*, 2015.
[14] D. E. Denning, "An intrusion-detection model," *IEEE Transactions on Software Engineering*, vol. SE-13, no. 2, pp. 222–232, 1987.