International Journal of Scientific Research and Engineering Development-– Volume 9 Issue 4, July – August 2026
Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

Hybrid Dynamic Security Policy Enforcer Using AI

Keerthana M
Dept of ISE
Don Bosco Institute of Technology
Bengaluru, India
keerthanam451@gmail.com

Dr. Ramya K V
Dept of ISE
Don Bosco Institute of Technology
Bengaluru, India
salmajabeen@dbit.co.in

Koushalya A Naik
Dept of ISE
Don Bosco Institute of Technology
Bengaluru, India
soumise23154@gmail.com

----------**********----------

Abstract:

Web-based systems are increasingly exposed to security threats, and traditional fixed security mechanisms are often unable to deal with new and constantly changing attack patterns. Most existing approaches depend either on predefined security rules or machine learning models, which limits their ability to identify and respond to complex threats in real time. This paper proposes a Hybrid Dynamic Security Policy Enforcer Using Artificial Intelligence that combines rule-based security policies with an unsupervised machine learning algorithm, Isolation Forest, to identify unusual activities and dynamically enforce security policies. The system continuously observes user login behavior, session activities, and network traffic, and assigns a risk score ranging from 0 to 100 for each interaction. When abnormal behavior is detected, the active session is automatically frozen, and an email verification alert is sent to the legitimate user. The system also uses an adaptive policy mechanism in which repeated violations result in increasingly strict actions, starting with a five-minute verification period for the first offense, followed by a two-minute period for the second offense, and a permanent block for the third offense. In addition, the system detects SQL injection and cross-site scripting attacks, monitors ICMP traffic, tracks IPv4 and IPv6 packets with geolocation information, and provides a live dashboard for administrators. An integrated attack simulator tests three common scenarios—brute-force, credential-stuffing, and bot-based attacks—to evaluate the system's response. The experimental results indicate that the proposed hybrid approach provides better overall security by combining anomaly detection, automated response, and adaptive policy enforcement compared with rule-based and AI-only methods.

----------**********----------

I. INTRODUCTION

The rapid growth of web applications and digital services has brought with it a significant rise in cybersecurity risks. Both organizations and individual users are now exposed to various types of attacks, including brute-force login attempts, credential stuffing, session hijacking, and automated bot activities. Conventional security methods, such as static firewalls and rule-based intrusion detection systems, often struggle to handle these changing threats because they depend on predefined rules and fixed limits that cannot respond effectively to new attack patterns in real time.

Machine learning techniques have become increasingly useful for identifying unusual behavior in digital environments. Among these, unsupervised methods such as Isolation Forest are well suited for security-related anomaly detection because they can work without labeled datasets and identify abnormal behavior based on patterns present in the data. However, relying only on AI-based detection can be challenging because such systems may not always provide immediate and clearly defined actions when a threat is identified, which is important for practical security applications.

To overcome these challenges, this paper introduces a Hybrid Dynamic Security Policy Enforcer Using Artificial Intelligence. The proposed approach combines a rule-based security layer with an AI-based anomaly detection component to create a more flexible and responsive security framework. The rule-based layer manages clearly identifiable events, including multiple failed login attempts, session expiration, and potentially harmful inputs, while the Isolation Forest model identifies behavioral abnormalities that may not be detected through predefined rules, such as unusual login times,

unexpected request rates, and suspicious patterns in user sessions.

One of the main features of the proposed system is its dynamic policy enforcement mechanism, which considers the previous attack history of a user when deciding how to respond to a new security incident. Unlike conventional systems that generally apply the same response to every event, the proposed approach increases the level of security enforcement after repeated violations. The response begins with email verification for the first offense, followed by a shorter verification period for the second offense, and finally permanent account blocking after the third offense. This continuous interaction between AI-based detection and rule-based enforcement allows the security mechanism to adapt according to the level of threat.

The system also provides a human-in-the-loop verification process, through which the legitimate user receives an email notification containing YES and NO options whenever suspicious behavior is detected. If the user confirms that the activity is genuine, the affected session is allowed to continue. If the user reports that the activity was not performed by them, the session is blocked immediately. When there is no response within the specified verification period, the system automatically blocks the session, ensuring that a potentially harmful activity does not remain active without supervision.

In addition, the proposed system performs network-level monitoring using ICMP ping analysis and supports real-time packet monitoring for both IPv4 and IPv6, along with IP-based geolocation. An administrative dashboard provides a live view of security incidents, calculated risk scores, and blocked sessions. An integrated attack simulator is used to reproduce three practical attack scenarios—brute force, credential stuffing, and bot activity—to evaluate the detection and enforcement mechanisms.

The rest of this paper is organized as follows. Section II discusses existing work related to anomaly detection, intrusion detection, and automated incident response. Section III explains the architecture of the proposed system.

Section IV describes the system flow and its major processing stages. Section V compares the proposed approach with existing security methods. Section VI discusses the algorithms implemented in the proposed system. Section VII presents the overall methodology followed in this work. Section VIII discusses the experimental results and their observations. Finally, Section IX summarizes the work and presents the conclusion.

II.RELATED WORK

Research on security in web applications and networked environments has expanded considerably as cyberattacks continue to become more sophisticated. As a result, researchers have investigated different combinations of machine learning, anomaly detection, rule-based security, and automated response techniques. This section discusses ten studies that are particularly relevant to the development and motivation of the proposed system.

Al-Shehari et al. [1] developed an insider-threat detection approach based on the Isolation Forest algorithm, particularly addressing the imbalance commonly present in security datasets. Their experimental results showed that the Isolation Forest approach achieved an F-score of 99% for anomaly detection, performing better than conventional supervised methods such as Logistic Regression, Naive Bayes, and K-Nearest Neighbors. However, the work was mainly designed for detecting threats from previously collected data and did not include continuous session monitoring or mechanisms for dynamically enforcing security policies.

Ferrag et al. [2] carried out a detailed review of intrusion detection systems based on deep learning techniques. Their study examined several neural network architectures, including Convolutional Neural Networks, Recurrent Neural Networks, and Long Short-Term Memory networks, for identifying malicious behavior in network environments. The findings showed that deep learning can recognize complicated attack patterns effectively. However, these techniques often require significant computing resources and large volumes of labeled data, which can make their implementation difficult in systems with limited computational capacity.

Syed et al. [3] introduced a dynamic access control framework based on the Zero Trust model for cloud computing environments. Their Trust-Based Access Control mechanism continuously assessed the trust level of users through a Deep Q-Network and modified access permissions according to changes in user behavior. The results showed that regularly updating user trust could lower the possibility of unauthorized access. However, the study concentrated mainly on cloud-based access control and did not

provide session-level security enforcement or an automated mechanism for responding to detected incidents.

Aldweesh et al. [4] examined different clustering methods for User and Entity Behavior Analytics to identify unusual activities from user activity logs. The researchers evaluated techniques such as K-Means, DBSCAN, and hierarchical clustering and considered factors including login frequency, access timing, and request volume. Their results indicated that analyzing user behavior patterns can help identify insider threats and potentially compromised accounts. However, the research primarily involved offline analysis of collected data and did not provide a mechanism for enforcing security policies immediately when suspicious behavior occurs.

Jamal et al. [5] presented a hyper-automation framework for Security Orchestration, Automation and Response in Security Operations Centers using Agentic Artificial Intelligence. The study showed that automating incident-handling procedures can reduce the time required to respond to security events when compared with manual intervention. Although the concept has similarities with the proposed system, its implementation focuses mainly on security operations at the organizational and network levels. It does not specifically provide session-level responses combined with direct verification from the legitimate user.

Liu et al. [6] investigated the use of machine learning for identifying abnormal network behavior and improving security defense mechanisms. Their work evaluated several unsupervised techniques, including Support Vector Machines, Isolation Forest, and Autoencoders, for detecting unusual activities within network traffic. The evaluated approaches produced effective results across different performance measures. However, the study was primarily concerned with network-level anomalies and did not consider application-level user behavior, active session management, or the dynamic modification of security policies.

Hussein Ali et al. [7] presented a detailed review of machine learning methods used in intrusion detection systems and discussed the important factors involved in their implementation. Their analysis covered different attack categories, including denial-of-service, brute-force, and remote-to-local attacks, while comparing signature-based, anomaly-based, and hybrid detection techniques. The study indicated that combining rule-based detection with anomaly detection can provide better security than relying on either approach alone. However, it did not examine progressive policy escalation, individual session control, or user-based verification during the response process.

Khader et al. [8] analyzed the shift from conventional perimeter-based security toward the Zero Trust security model and examined how artificial intelligence and machine learning can support this transition. The study discussed the use of behavioral analysis and real-time anomaly detection to improve automated security decisions by continuously evaluating users against predefined trust profiles. The findings suggested that AI-supported Zero Trust approaches can strengthen real-time threat handling. However, the research remained largely conceptual and did not present a practical implementation for session-level enforcement or progressively stricter security policies.

Chen et al. [9] introduced ZT-SDN, an automated framework designed to learn and enforce network access policies within Software-Defined Networks using Zero Trust principles. The framework applied unsupervised learning to understand normal behavioral patterns and automatically make access-control decisions. Its evaluation showed that the approach could identify abnormal network access using real-world datasets. However, ZT-SDN mainly operates at the network layer and does not address application-level session monitoring, interaction-based risk scoring, email verification by legitimate users, or progressive policy enforcement according to repeated attack history.

Liu et al. [10] introduced the Isolation Forest algorithm as an efficient unsupervised method for detecting abnormal data points. The basic idea of the algorithm is to separate observations through random divisions of the feature space, where unusual data points generally become isolated with fewer divisions than normal observations. This property allows anomalies to be identified without depending on labeled training data. The study forms the theoretical foundation for the Isolation Forest-based anomaly detection component incorporated into the proposed security system.

Research Gap

The review of these studies identifies several limitations that motivate the development of the proposed system. Many existing solutions depend either on predefined security rules or machine learning models separately instead of combining their strengths

within a single framework. In addition, anomaly detection approaches generally concentrate on identifying suspicious behavior without dynamically changing security policies according to previous attack history. Automated response solutions are often designed for network or organizational environments rather than controlling individual user sessions and involving the legitimate user in the verification process. Another limitation is the absence of a progressive enforcement mechanism in which repeated violations lead to increasingly strict security actions. The proposed Hybrid Dynamic Security Policy Enforcer aims to overcome these limitations by bringing together AI-based anomaly detection, rule-based security checks, adaptive policy escalation, human verification, and automated incident response within a unified real-time framework.

III. PROPOSED SYSTEM

The proposed Hybrid Dynamic Security Policy Enforcer Using Artificial Intelligence follows a two-service architecture that integrates a Node.js and Express backend with a Python and Flask-based AI inference service. These services exchange data through REST APIs, allowing the system to perform anomaly analysis and apply security policies while user interactions are taking place. The complete architecture consists of six major functional components: the authentication layer, AI detection engine, dynamic policy enforcement engine, session management layer, incident response module, and administrative monitoring dashboard.

A. Authentication Layer

The authentication layer manages incoming login requests and provides the initial security check using predefined rules. Whenever credentials are submitted, the system examines the input fields for potentially harmful patterns, such as SQL injection commands and cross-site scripting payloads, using predefined regular expressions. Any request identified as malicious is rejected immediately and recorded with a risk score of 100. For valid input, the submitted credentials are compared with the user database while failed login attempts are tracked against the requesting IP address. Three successive failures cause the system to start a temporary cooldown period, preventing additional login attempts during that interval. If the total number of failed attempts reaches five, the corresponding account is locked and an email notification is sent to the registered address, allowing the account owner to choose whether to unlock the account or leave it blocked.

B. AI Detection Engine

The AI detection engine operates within the Python Flask service and applies the Isolation Forest algorithm to identify abnormal behavior without requiring labeled attack data. The algorithm repeatedly divides the feature space into random partitions and observes how easily individual observations become separated from the remaining data. Observations that are isolated with fewer partitions are treated as potential anomalies because they differ from normal behavior patterns. The model is trained using normal login and session activity and considers three main features: the login hour representing when the user attempted to log in, the number of previous failed attempts, and the total number of requests generated during the active session. For every interaction, Isolation Forest generates an anomaly value that is transformed into a risk score ranging from 0 to 100 through a custom normalization method. Scores above 50 are treated as suspicious, while values close to 100 represent a stronger indication of abnormal activity. The AI service analyzes every login request and session activity request, allowing user behavior to be monitored continuously in real time.

C. Dynamic Policy Enforcement Engine

The dynamic policy enforcement engine represents one of the main features of the proposed security framework. Instead of applying an identical response to every security event, the engine stores the previous anomaly history of each user and uses it to determine the appropriate enforcement level. Three policy stages are defined by the system. At the first detected anomaly, the normal policy is activated, sending an email verification request and starting a five-minute automatic blocking timer. When another anomaly is detected for the same user, the strict policy is applied, generating a warning email and reducing the verification period to two minutes. If a third anomaly is identified, the permanent policy is triggered, immediately ending the active session and permanently blocking the account without another email verification step. The repeated anomalous behavior is therefore treated as sufficient evidence of potentially malicious activity. By connecting the AI detection results with the rule-based response mechanism, the system can continuously adjust its

security behavior according to the user's attack history.

D. Session Management Layer

The session management component observes user activity after successful authentication and applies security controls at the individual session level. Information maintained for each session includes the login timestamp, login hour, number of requests, most recent activity time, and current frozen status. If a session remains inactive for fifteen minutes, it is automatically terminated by the timeout mechanism. Whenever the AI engine identifies suspicious session behavior, the session is frozen immediately so that no additional actions can be performed until verification is completed. If the user selects YES through the email notification, the session is restored and the request counter is reset so that normal activity can continue. Selecting NO causes the session to be removed immediately. When the user fails to respond within the time period specified by the active policy, the system automatically deletes the session and records the event as an automatic blocking incident.

E. Incident Response Module

The incident response module is responsible for automated notifications and recording the details of security-related events. Whenever suspicious behavior is identified, the system generates an HTML-formatted email and sends it to the registered address of the affected user. The notification contains relevant information such as the user's email address, calculated risk score, action taken, request count, and event timestamp. It also provides YES and NO buttons that connect to the backend endpoints responsible for confirming or blocking the session. The server's IP address is identified automatically during runtime so that the links included in the email use the appropriate server address in different network environments. Security activities such as login attempts, anomaly detections, session confirmations, blocked sessions, automatic blocks, and policy changes are recorded in both an in-memory activity store and a persistent JSON log file. This provides a detailed record that can be used for tracking and auditing security incidents.

F. Administrative Monitoring Dashboard

The administrative monitoring dashboard offers administrators a real-time view of activities occurring throughout the security system through a web interface that updates at two-second intervals. It presents overall statistics such as the number of recorded events, successful authentication attempts, failed logins, and detected anomalies. The live activity section displays recent security events along with event-type indicators, affected users, risk scores represented through visual progress bars, and the corresponding enforcement action. The dashboard also contains a network packet monitoring section that provides information such as the source IP address, whether the address uses IPv4 or IPv6, geographical location including city and country, communication protocol, port number, packet size, and packet status. In addition, an ICMP monitoring component evaluates network health by tracking latency, packet loss, and availability for the local machine, gateway, and external DNS servers. This combination provides administrators with visibility into both application-level security events and network-level conditions.

G. Attack Simulator

A dedicated attack simulator is included in the system to reproduce common attack situations and evaluate the effectiveness of the proposed security mechanisms. It supports three scenarios: brute-force attacks, credential stuffing, and automated bot activity. During the brute-force test, twenty rapid login requests are generated using a collection of commonly used passwords, allowing the rule-based protection to demonstrate account blocking after five unsuccessful attempts. The credential-stuffing scenario uses several username and password combinations obtained from commonly available leaked-credential lists to test whether the AI component can recognize unusual login behavior across different accounts. For the bot-activity scenario, a legitimate account is used to generate twenty session requests within a short period. This activity is intended to trigger anomaly detection and demonstrate the complete response sequence, including session freezing, email-based verification, and dynamic security policy enforcement.

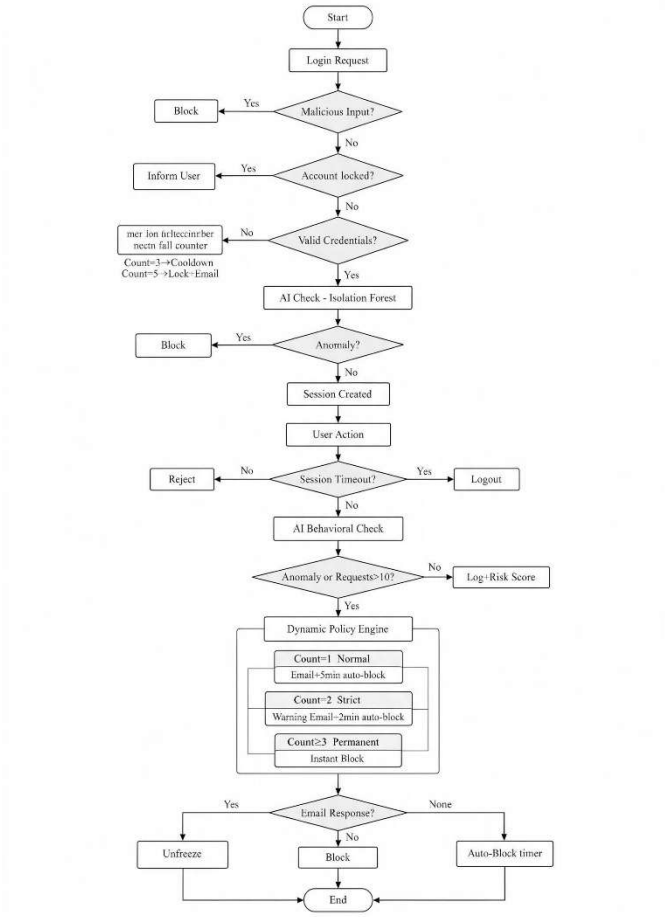

Fig.3.1 Architecture of hybrid dynamic policy enforcer using AI system displaying the data flow

IV.METHODOLOGY

The methodology adopted for the proposed Hybrid Dynamic Security Policy Enforcer Using Artificial Intelligence is organized as a systematic sequence that combines data preparation, machine learning, application development, security validation, and real-time policy enforcement. Each stage contributes to the overall functioning of the system, beginning with the design of the system architecture and preparation of the training data, followed by behavioral analysis, authentication, session monitoring, dynamic response, incident handling, network observation, dashboard visualization, and attack-based validation.

A. System Architecture Design

The proposed framework is designed using a two-service microservice architecture in which security enforcement and artificial intelligence processing are handled separately. The security-oriented backend is developed with Node.js and Express and is responsible for user authentication, session handling, policy enforcement, packet observation, and administrative dashboard operations. A separate AI service is developed using Python and Flask to manage machine learning model initialization, anomaly prediction, risk-score calculation, and ICMP-based network monitoring. Communication between these two services takes place through HTTP-based REST APIs. During operation, the Node.js server collects the required behavioral information and sends the resulting feature vector to the Flask service, which performs the anomaly analysis and returns the prediction and risk information. This division allows the machine learning component to be modified or replaced without making major changes to the security backend. It also supports scalability because additional instances of either service can be deployed independently when required. For demonstration and testing, the user interface consists of static HTML, CSS, and JavaScript resources that are delivered directly through the Node.js server, allowing the complete application to operate through a single deployment environment.

B. Dataset Preparation and Model Training

The Isolation Forest model is developed using a dataset representing normal login and session behavior rather than relying on labelled attack samples. Each record contains three main behavioral attributes: login hour, which indicates the time of day when the authentication activity occurs and ranges from 0 to 23; failed attempt count, which records previous unsuccessful login attempts associated with the requesting IP address; and request count, which represents the number of requests generated during the user's active session. The training data is designed to capture realistic legitimate behavior, such as authentication during common working hours, a small number of failed attempts caused by occasional password mistakes, and moderate request activity during ordinary system usage. The Isolation Forest implementation provided by Scikit-learn is used for training, with the contamination value selected according to the expected proportion of unusual activities in the monitored environment. After the training process is completed, the resulting model is stored as a pickle file. The Flask AI service loads this serialized model when the application starts, allowing predictions to be generated immediately during operation without repeatedly performing the training process.

C. Feature Extraction and Risk Score Computation

Whenever a user performs a login operation or generates activity within an existing session, the system collects the three behavioral attributes required by the machine learning model. These values are organized into a feature vector and transmitted from the Node.js backend to the Flask-based AI service. The trained Isolation Forest model processes the received information and produces both an anomaly prediction and a raw anomaly score. Since the raw output is not directly convenient for interpreting security severity, it is converted into a normalized risk value ranging from 0 to 100 using the custom scaling mechanism defined for the proposed system. The resulting risk score and anomaly classification are then returned to the Node.js backend, where they contribute to the selection of the appropriate security response. The calculated information is also stored in the activity records so that administrators can review the detected behavior and corresponding risk level through the dashboard.

D. Authentication and Input Validation

When authentication information is submitted, the system initially examines the email and password fields using the input validation mechanism to identify possible SQL injection and cross-site scripting patterns. Any request containing a recognized malicious pattern is rejected immediately and is not allowed to proceed to the subsequent authentication stages. When the submitted values pass this validation step, the system checks whether the account is currently locked and evaluates the number of unsuccessful attempts associated with the requesting IP address before comparing the credentials with the user database. If the credentials are valid, the request is passed to the AI service so that the user's login behavior can be evaluated. When the machine learning model identifies the activity as normal, the system creates a new session and permits the user to access the application. Conversely, if the login behavior is classified as anomalous, access is denied and the incident is recorded together with the calculated risk score.

E. Session Monitoring and Anomaly Detection

Following successful authentication, the system continuously observes user interactions through the session activity endpoint. Each action generated during an active session is handled by the session management component, which first verifies whether the session has been frozen or has exceeded its allowed lifetime. Once these checks are completed, the session request counter is updated and the relevant behavioral features are forwarded to the AI inference service for further analysis. The returned anomaly result and risk value are recorded in the activity history and are used to determine whether stronger security controls need to be applied. In the proposed implementation, the dynamic policy enforcement mechanism is activated when either the number of requests becomes greater than 10 or the AI model identifies the activity as anomalous. Therefore, suspicious behavior can trigger enforcement immediately without waiting for the request count threshold when an anomaly is detected earlier.

F. Dynamic Policy Enforcement

When suspicious behavior activates the policy enforcement mechanism, the system retrieves the user's accumulated attack information from the in-memory attack history store. The recorded attack count is then increased to include the newly identified incident, and the appropriate policy level is selected according to the updated value. Under the NORMAL policy, the active session is temporarily frozen, an email containing information about the suspicious activity and YES/NO response options is sent to the user, and a five-minute timer is started. When the STRICT level is reached because of a repeated violation, the current session is again frozen, but the user receives a warning notification indicating that another suspicious incident has occurred, while the waiting period is reduced to two minutes. Once the PERMANENT level is reached, the system treats the repeated anomalous behavior as sufficient evidence of a serious threat and immediately terminates the active session without requesting email confirmation. This progressive approach allows the system to increase enforcement strength according to the user's history rather than applying the same response to every incident.

G. Incident Response and Email Verification

The incident-response component generates an HTML-formatted security notification containing information about the detected event, along with YES and NO buttons that allow the user to respond to the alert. The links associated with these controls contain the server IP information required by the corresponding backend endpoints. Emails are

delivered through the Nodemailer library using the Gmail SMTP service. If the user selects YES, the confirmation endpoint restores the frozen session, clears the current request count, marks the alert as confirmed, and stops the automatic blocking timer. Selecting NO invokes the blocking endpoint, which immediately removes the active session and records the alert as blocked. If the user does not respond before the applicable policy timeout expires, the automatic timer executes the blocking procedure, terminates the active session, changes the alert status to timed out, and records the incident as an automatic block caused by the absence of a response.

H. Network Monitoring

Network observation is performed at both the application and monitoring-service levels to provide additional information about system activity. At the application layer, the Node.js backend records each incoming request as a packet-related event containing information such as the source and destination IP addresses, communication protocol, port number, packet size, event category, status, and geographic information obtained through a third-party IP geolocation service. The system determines whether an address belongs to IPv4 or IPv6 according to its format, while simulated IPv6 addresses are generated for attacker-associated records to demonstrate IPv6 monitoring functionality. At the network-monitoring layer, the Flask AI service runs a background process that sends a ping request to the localhost address every two seconds using the operating system's ping utility. The resulting measurements include response latency, packet-loss information, and the current online or offline state. In addition, the administrative interface provides a dedicated ICMP testing endpoint through which administrators can manually check connectivity to the localhost, gateway, and external DNS server targets.

I. Administrative Dashboard

The administrative dashboard provides a centralized interface for observing the current security state of the application and is implemented as a static HTML page that communicates with the backend at two-second intervals. During each update cycle, the dashboard retrieves the latest statistical information, activity entries, and network packet records. Summary information includes the overall number of recorded events, successful authentication attempts, unsuccessful logins, and anomalies identified by the system. A live activity section presents recent security events together with visual indicators for risk levels, enforcement outcomes, and anomaly information. The packet-monitoring section provides current network records and identifies IPv4 and IPv6 traffic along with available geolocation information and status indicators. A separate ICMP health section displays the connectivity state of monitored targets, including their response latency, packet-loss percentage, and accumulated ping count, allowing administrators to obtain a real-time overview of both application-level security events and network conditions.

J. Attack Simulation and Validation

The proposed system is evaluated using an integrated attack simulator designed to reproduce three representative security scenarios. In the brute-force simulation, twenty rapid authentication attempts are generated against a known user account using a predefined collection of commonly used passwords. This activity is intended to activate the failed-attempt monitoring mechanism and demonstrate account locking together with email-based notification. The credential-stuffing simulation uses six different username and password combinations based on commonly observed leaked-credential patterns to evaluate whether the AI component can recognize unusual login behavior occurring across multiple accounts. The final scenario represents bot-like activity, where a legitimate user first establishes a valid session and then generates twenty session-activity requests within a short period. This rapid sequence is designed to trigger the anomaly detection mechanism and demonstrate the complete response process, including session freezing, email verification, and progressive policy escalation. Together, these simulations provide an end-to-end validation of the rule-based controls, AI-based detection, incident response, and dynamic enforcement capabilities of the proposed security framework.

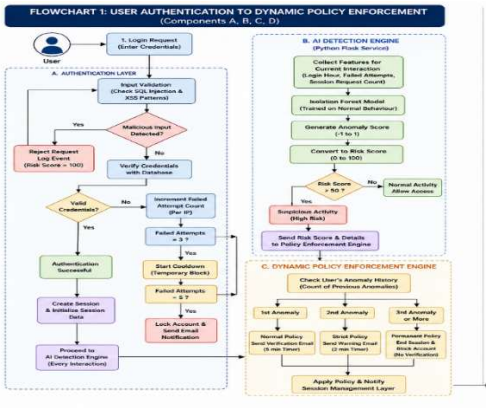

Fig.4.1 flowchart: User authentication to dynamic policy enforcement

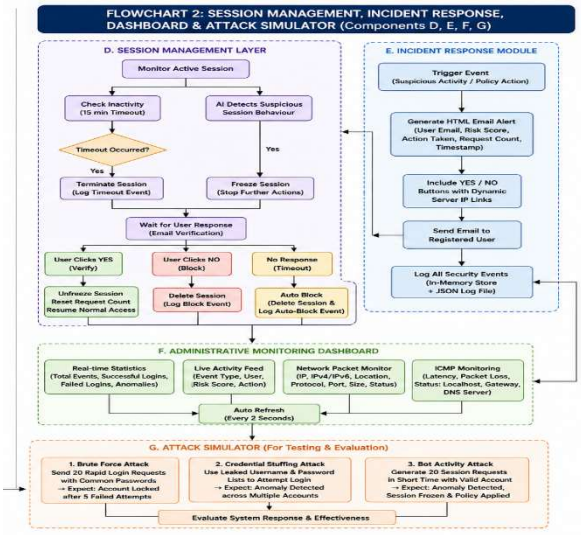

Fig.4.2 flowchart: session management, response dashboard and attacking simulator

V. SYSTEM FEATURES

The proposed Hybrid Dynamic Security Policy Enforcer Using Artificial Intelligence combines multiple security features to provide continuous monitoring and real-time protection against suspicious activities.

Authentication Security: The system provides multiple layers of login protection. It detects SQL injection and cross-site scripting attempts during input validation, tracks repeated login failures, introduces a cooldown after three unsuccessful attempts, and locks the account after five failed attempts. An email-based recovery option is also provided so that only the genuine account owner can regain access.

AI-Powered Anomaly Detection: The system applies the Isolation Forest algorithm to identify unusual user behavior in real time. For each user interaction, three behavioral parameters — login time, number of failed attempts, and session request count — are considered. These values are converted into a risk score ranging from 0 to 100, where a higher score represents a stronger deviation from normal behavior.

Dynamic Policy Enforcement Engine: A major feature of the proposed system is its ability to increase security restrictions according to the user's anomaly history. Instead of applying a fixed response to every suspicious event, the system gradually strengthens the policy. The first anomaly activates a normal policy with a five-minute verification period, the second applies a strict policy with a two-minute period, while the third results in a permanent block without additional verification.

Session Management: All active sessions are continuously monitored by the system. A session is automatically terminated after fifteen minutes of inactivity, while the detection of suspicious behavior immediately freezes the session. A polling mechanism checks for user confirmation within three seconds and restores access when the activity is verified as legitimate.

Human-in-the-Loop Verification: When unusual activity is identified, the registered user receives an HTML-formatted email describing the suspicious event along with YES and NO response options. This allows genuine users to confirm their activity while helping the system prevent unauthorized access.

Automated Incident Response: The system follows an automated security response sequence consisting of detection, alert generation, session freezing, user verification, and policy escalation. No administrator intervention is required during this process. If the user does not respond within the specified time, the system automatically blocks the suspicious activity.

Network Monitoring: Network behavior is observed at both the application and network levels. At the application level, incoming requests are monitored along with IPv4 or IPv6 information, geolocation, protocol, port number, and packet size. In addition, ICMP monitoring checks the latency, packet loss, and connectivity status of the localhost, gateway, and external DNS server.

Administrative Dashboard: The system provides a live administrative dashboard that updates every two seconds. It presents current security statistics, a color-coded activity feed with risk scores, network packet information, and ICMP health status, allowing administrators to monitor the overall security condition from a single interface.

Attack Simulator: For testing and demonstration, the

system provides an integrated attack simulator that reproduces three common attack situations: brute-force attacks, credential stuffing, and bot-based activity. These simulations help verify whether the proposed security mechanisms respond correctly under different attack conditions.

VI. EXPECTED RESULTS AND OUTCOMES

The proposed Hybrid Dynamic Security Policy Enforcer is expected to achieve the following results after complete implementation, evaluation, and testing.

Improved Threat Detection Accuracy: The combination of rule-based techniques and the Isolation Forest algorithm is expected to provide broader threat detection than using either method alone. The rule-based layer should effectively identify predefined attacks such as brute-force attempts and SQL injection, while the AI component is expected to identify less obvious changes in user behavior that may indicate suspicious activity.

Real-Time Risk Quantification: Each user interaction is expected to generate a risk score ranging from 0 to 100, giving administrators a clear numerical indication of the associated security risk. As the number of session requests increases, the risk score is expected to rise accordingly, helping identify possible bot activity or session misuse at an early stage.

Reduced Incident Response Time: The automated response mechanism is expected to considerably reduce the time required to handle security incidents. Instead of depending on manual administrator action, the system can automatically freeze suspicious sessions and send alerts immediately after detecting an anomaly. Depending on the applied policy, the auto-block mechanism is expected to stop unaddressed threats within approximately two to five minutes.

Adaptive Policy Enforcement: The dynamic escalation mechanism is expected to provide stronger protection against users who repeatedly generate suspicious activity. Each subsequent violation results in a stricter security response. If an attacker continues to trigger anomalies across three or more sessions, the system is expected to permanently block further access without providing another verification period.

Preserved Legitimate User Access: The human-in-the-loop verification process is expected to reduce the effect of false alarms on genuine users. When legitimate activity is flagged, the user can verify it through the email notification and regain access within a short period. This approach helps maintain security without unnecessarily interrupting normal user activities.

Network-Level Visibility: The packet monitoring and ICMP-based monitoring components are expected to give administrators a real-time view of important network conditions. This information can help identify unusual network behavior, service disruption, or increased activity that may be associated with attacks such as denial-of-service attempts or network scanning.

Comprehensive Audit Trail: Security-related activities such as login attempts, detected anomalies, policy changes, session confirmations, blocked sessions, and automatic blocking events are expected to be recorded in real time. Maintaining these records in both the in-memory activity store and persistent JSON log files will provide useful information for incident investigation, system monitoring, and future compliance requirements.

Validated Attack Resistance: The system is expected to be evaluated using three attack scenarios: brute-force attacks, credential stuffing, and bot activity. Successful detection and response within the specified policy time limits would demonstrate that the hybrid approach can effectively identify and control different types of suspicious behavior under realistic testing conditions.

TABLE I -EXPECTED OUTCOMES

Area	Expected Outcome
Threat Detection	Wider and more accurate detection using rule-based and AI techniques
Risk Assessment	Real-time risk scoring from 0–100 for every user interaction
Incident Response	Faster automated detection, alerting, session freezing, and blocking
Policy Enforcement	Progressive security policies based on repeated anomalous activity
User Verification	Reduced false positives through email-based user confirmation
Network Monitoring	Real-time visibility into network health and suspicious activity
Security Audit	Complete logging of security events for analysis and reporting
Attack Resistance	Effective detection and response to brute force, credential stuffing, and bot attacks

VII. FUTURE SCOPE

The proposed Hybrid Dynamic Security Policy Enforcer provides a foundation for adaptive security using AI, but its capabilities can be further improved through several possible extensions and future developments.

Enhanced Feature Extraction: The present system relies on three behavioral parameters for detecting anomalies. In future versions, more user-related characteristics such as typing patterns, mouse movements, device fingerprints, location consistency, and usual login times can be considered. Using these additional features may help identify advanced attacks that are designed to appear similar to normal user activity.

Online Learning and Model Adaptation: At present, the Isolation Forest model is trained using a fixed dataset containing normal behavioral patterns. A future implementation could allow the model to learn continuously from newly confirmed normal and abnormal events. This would help the system adjust to changes in user behavior and respond better to newly emerging attack patterns.

Multi-Factor Authentication Integration: The existing email-based verification process can be expanded by adding other authentication methods. These may include SMS-based one-time passwords, mobile push notifications, and hardware security keys. Such additional verification methods can provide stronger identity validation when a high-risk activity is detected.

Configurable Policy Thresholds: Currently, the anomaly thresholds and policy timeout values are predefined within the system configuration. In the future, these values could be made adjustable through the administrative dashboard. This would allow security administrators to modify the level of enforcement according to the requirements and risk conditions of their organization.

Real-Time Packet Capture Integration: The current network monitoring module mainly records application-level packet information. Future development can incorporate a packet capture tool such as PyShark, which works with the Wireshark engine, to inspect actual network traffic. Combining this with the existing monitoring approach would provide better visibility at both application and network levels.

Machine Learning-Based Policy Recommendation: Another possible improvement is the addition of a machine learning module that recommends suitable security policies automatically. By studying previous attack incidents from different users, the system could suggest appropriate thresholds, cooldown periods, and escalation conditions based on user groups and their access privileges.

Distributed Deployment Support: The current system is mainly intended for deployment on a single server. Future versions could be designed for distributed environments involving multiple servers or cloud platforms. A centralized policy system along with synchronized attack history would allow security decisions to remain consistent across different nodes.

Mobile Application Integration: A dedicated mobile application could be introduced to provide security notifications and verification requests directly to users. Push notifications could make the response process faster and more convenient than relying only on email-based alerts, especially when immediate confirmation is required.

Integration with SIEM Platforms: The generated security logs and risk scores can be connected with existing Security Information and Event Management platforms such as Splunk and IBM QRadar. This would allow the proposed system to work as an anomaly detection and policy enforcement component within a larger enterprise security infrastructure.

Graph-Based Behavioral Analysis: Future research could also investigate graph-based methods, including graph neural networks, to represent connections between users, sessions, and activities. Such an approach may help identify coordinated attacks involving multiple accounts or sessions that could remain undetected when each session is analyzed separately.

VIII. CONCLUSION

This work presented the Hybrid Dynamic Security Policy Enforcer Using Artificial Intelligence, a real-time security framework that combines predefined security rules with unsupervised machine learning to identify and respond to different forms of cyber threats. The system integrates multiple security mechanisms within a single workflow, allowing suspicious activity to be detected, evaluated, logged, and subjected to increasingly stronger enforcement according to the user's attack history. Through this design, the framework provides a more adaptive approach to security policy management than systems that rely on a single detection technique.

The proposed architecture combines the predictable behavior of rule-based controls with the behavioral analysis capability of the Isolation Forest model. The rule-based layer provides immediate protection against recognizable threats, including brute-force attempts, SQL injection, cross-site scripting, and inactive-session misuse. At the same time, the AI component examines behavioral characteristics such as login timing and session request frequency to identify activity that differs from the learned normal pattern. Bringing both approaches into the same security pipeline enables the system to handle deterministic attacks and less obvious behavioral deviations without depending entirely on either rules or machine learning.

A major feature of the framework is its cumulative policy enforcement mechanism. Instead of applying an identical response to every detected anomaly, the system maintains the user's previous attack history and progressively increases the enforcement level from NORMAL to STRICT and finally to PERMANENT when repeated violations occur. This feedback-based design allows the security response to become stronger as suspicious behavior continues, creating an adaptive defense mechanism that is more suitable for persistent attacks than a fixed-response security policy.

The human-verification mechanism also provides a balance between security and legitimate user accessibility. When an anomaly is identified, the system can temporarily freeze the affected session and send an email containing YES and NO response options. A legitimate user can therefore confirm the activity and regain access, while an unauthorized user can be prevented from continuing. The automatic blocking mechanism provides an additional safeguard by terminating the session when no response is received within the specified period. Consequently, the system does not depend entirely on either human intervention or automatic decision-making.

The experimental evaluation demonstrated that the major components of the framework operated according to their intended behavior. The brute-force protection successfully restricted repeated password attempts through progressive cooldown and account-locking mechanisms, while the email-based recovery process allowed legitimate users to restore access. The AI component produced progressively higher risk values as session activity became more intensive and identified abnormal request patterns during the bot simulation. Session freezing and email verification prevented further use of suspicious sessions while still providing a controlled recovery mechanism. The dynamic escalation component correctly strengthened the response across repeated anomaly events, eventually applying permanent blocking on the third violation. The automatic blocking mechanism also terminated sessions when required response periods expired under both normal and strict policies.

The network-monitoring features extended the visibility of the framework beyond authentication and session activity. The ICMP monitoring service successfully measured connectivity, latency, packet loss, and availability for the selected network targets, while the packet-monitoring component recorded incoming requests and provided IPv4/IPv6 classification and geolocation information. These records were presented through the administrative dashboard together with security events and risk information, giving administrators a centralized view of the application's current security condition.

The comparative evaluation indicates that the proposed framework brings together a wider range of capabilities than the individual approaches considered in the related work. In particular, the system combines rule-based protection, AI-based anomaly detection, cumulative policy escalation, session freezing and restoration, email-based human verification, automatic incident response, IPv4 and IPv6 network observation, and a unified real-time administrative dashboard. The integration of these functions within one framework makes the proposed approach more comprehensive and adaptable for real-time security monitoring.

Future Work

Several improvements can be considered to extend the capabilities of the current implementation. The Isolation Forest model could be supplied with additional behavioral attributes, including typing characteristics, mouse interaction patterns, device fingerprints, and consistency of geographic information, which may improve detection of more sophisticated attacks. Instead of depending on a fixed training dataset, an online-learning approach could be introduced so that the model periodically incorporates verified normal and anomalous events and adapts to changes in legitimate user behavior. The policy thresholds could also be exposed through the administrative interface, allowing security administrators to adjust enforcement levels according to the requirements and risk profile of a particular deployment. The existing email-confirmation

mechanism could be expanded with stronger authentication options, including one-time passwords and push-based approvals, particularly for high-risk events. Network visibility could be further improved by integrating a packet-capture framework such as PyShark, enabling more detailed inspection of network traffic in addition to the existing application-level monitoring. Another possible extension is the development of a machine-learning-based policy recommendation component capable of examining historical incidents and suggesting suitable enforcement thresholds for different categories of users and access levels.

Final Conclusion

In conclusion, the proposed Hybrid Dynamic Security Policy Enforcer Using Artificial Intelligence demonstrates the potential of combining rule-driven protection, unsupervised anomaly detection, and adaptive policy escalation within a single security framework. The experimental results show that the system can identify predefined attacks as well as unusual behavioral patterns, respond to detected incidents in real time, and increase enforcement strength when repeated violations occur. By incorporating automated protection, controlled human verification, session-level enforcement, network monitoring, and centralized visualization, the framework provides a practical foundation for developing more adaptive cybersecurity policy enforcement systems.

REFERENCES

[1] T. Al-Shehari, M. Al-Razgan, T. Alfaikh, R. A. Alsowail and S. Pandiaraj, "Insider Threat Detection Model Using Anomaly-Based Isolation Forest Algorithm," IEEE Access, vol. 11, pp. 118170–118185, 2023, doi: 10.1109/ACCESS.2023.3326750.

[2] M. A. Ferrag, L. Maglaras, A. Ahmim, M. Derdour, and H. Janicke, "RDTIDS: Rules and Decision Tree-Based Intrusion Detection System for Internet of Things Networks," Future Internet, vol. 12, no. 3, p. 44, 2020, doi: 10.3390/fi12030044.

[3] T. Syed, A. Alzahrani, S. Jan, M. S. Syed, A. Shamim, and T. Choudhury, "Zero-Trust Based Dynamic Access Control for Cloud Computing," Cybersecurity, 2024, doi: 10.1186/s42400-024-00320-x.

[4] A. Aldweesh, A. Derhab, and A. Z. Emam, "A Comprehensive Investigation of Clustering Algorithms for User and Entity Behavior Analytics," Frontiers in Big Data, vol. 7, 2024, doi: 10.3389/fdata.2024.1375818.

[5] A. Jamal et al., "Toward Robust Security Orchestration and Automated Response in Security Operations Centers with a Hyper-Automation Approach Using Agentic Artificial Intelligence," Mathematics, vol. 13, no. 14, p. 2315, 2025, doi: 10.3390/math13142315.

[6] R. Liu, J. Shi, X. Chen, and C. Lu, "Network Anomaly Detection and Security Defense Technology Based on Machine Learning," Computers and Electrical Engineering, vol. 118, 2024, doi: 10.1016/j.compeleceng.2024.109381.

[7] A. H. Ali, M. Charfeddine, B. Ammar, B. B. Hamed, F. Albalwy, A. Alqarafi, and A. Hussain, "Unveiling Machine Learning Strategies and Considerations in Intrusion Detection Systems: A Comprehensive Survey," Frontiers in Computer Science, vol. 6, 2024, doi: 10.3389/fcomp.2024.1387354.

[8] M. Khader, M. Fadul, K. Sheldon, and D. Adjeroh, "Zero Trust Security: A Comprehensive Analysis," arXiv preprint arXiv:2401.09575, 2024.

[9] A. Chen, F. Civerchia, K. Kondepu, L. Valcarenghi, and P. Castoldi, "ZT-SDN: Automated Zero Trust Framework for Learning and Enforcing Network Access Control," arXiv preprint arXiv:2411.15020, 2024.

[10] G. Madhukar Rao and D. Ramesh, "Web Traffic Anomaly Detection Using Isolation Forest," Informatics, vol. 11, no. 4, p. 83, 2024, doi: 10.3390/informatics11040083.

