International Journal of Science, Strategic Management and Technology
Volume **02** Issue **05** May-**2026** | ISSN: **3108-1762 (Online)** | Impact Factor: 3.8
An International, Peer-Reviewed, Open Access Scholarly Journal Indexed in recognized academic databases

Web-Based Intrusion Detection System for Login Attack Detection using Deep Learning Techniques

Muthupriya M[1] · Dr. A. Angel Cerli[2]

[1]B.Sc. Computer Science, Dept. of CS & IT, VISTAS, Chennai | [2]Assistant Professor, Dept. of CS & IT, VISTAS, Chennai

Vels Institute of Science, Technology and Advanced Studies (VISTAS), Pallavaram, Chennai – 600117, India

https://doi.org/10.55041/ijsmt.v2i5.047

Cite this Article: M, M. (2026). Web-Based Intrusion Detection System for Login Attack Detection using Deep Learning Techniques. International Journal of Science, Strategic Management and Technology, *02*(05). https://doi.org/10.55041/ijsmt.v2i5.047

Abstract

The rapid proliferation of web applications and internet-based services has significantly heightened the need for robust authentication and intrusion-prevention mechanisms. Among the most critical cybersecurity challenges are unauthorized login attempts and credential-based attacks, including brute force, dictionary, and credential stuffing techniques. Traditional security controls — password policies, CAPTCHA, and signature-based intrusion detection systems — have proven insufficient against these evolving threats. This paper proposes and evaluates a Web-Based Intrusion Detection System (IDS) that leverages deep learning to monitor, analyse, and detect suspicious login activities in real time. The system captures multidimensional authentication features — username, IP address, login frequency, geolocation, device fingerprint, and temporal patterns — and processes them through Artificial Neural Network (ANN) and Long Short-Term Memory (LSTM) models. Experimental results demonstrate that the deep learning-based IDS achieves superior detection accuracy, low false-positive rates, and sub-second classification latency, outperforming conventional rule-based approaches. The proposed system is designed as a modular, scalable architecture comprising frontend, backend, database, and AI inference components, making it applicable to banking, e-commerce, and enterprise environments.

Keywords: *Intrusion Detection System, Login Attack Detection, Deep Learning, Brute Force Attack, Credential Stuffing, LSTM, ANN, Web Security, Anomaly Detection, Cybersecurity.*

I. INTRODUCTION

Web applications have become indispensable in contemporary life, serving as gateways for online banking, e-commerce, healthcare, education, and government services. The login interface is universally the primary access control point; consequently, it is also the most actively targeted surface by malicious actors. Cyberattacks against authentication systems are escalating in frequency and sophistication, with attackers deploying automated tools capable of executing thousands of credential-testing attempts per minute [1].

Common offensive techniques include brute force attacks — systematic enumeration of password combinations — credential stuffing, which exploits leaked credential databases to test reuse across platforms, and dictionary attacks that leverage ordered lists of probable passwords. These methods are frequently orchestrated by botnets, making them difficult to mitigate with static countermeasures such as account lockout policies or CAPTCHA challenges [2].

Conventional Intrusion Detection Systems (IDS) operate on predefined signature databases or threshold-based rules. While effective against known attack vectors, they exhibit fundamental limitations when confronted with previously unseen or gradually evolving attack patterns. The integration of machine learning and, more recently, deep learning into IDS architectures addresses these shortcomings by enabling data-driven, adaptive threat detection [3].

This paper proposes a Web-Based IDS specifically designed for login attack detection. The system continuously monitors every authentication attempt, extracts behavioural features, and applies deep learning models — ANN and LSTM — trained on historical login data to classify each attempt as benign or malicious. Upon detecting an anomaly, the system invokes automated response actions, including adaptive CAPTCHA enforcement, temporary IP blocking, and administrator alerting. The remainder of this paper is structured as follows: Section II reviews related work; Section III presents the system architecture; Section IV describes the deep learning methodology; Section V covers implementation; Section VI reports experimental results; and Section VII concludes with future directions.

II. LITERATURE REVIEW

The evolution of intrusion detection has proceeded through three broad generations: signature-based, anomaly-based, and hybrid systems. Early IDS frameworks, most notably Denning's seminal 1987 intrusion detection model [4], introduced statistical profiling of user activity. Subsequent signature-based systems excelled at detecting known threats but required continual manual signature updates and were blind to novel attacks [5].

Machine learning was introduced to IDS research to overcome these limitations. Decision Trees, Support Vector Machines (SVM), K-Nearest Neighbours (KNN), and Naïve Bayes classifiers demonstrated substantially improved detection rates on benchmark datasets such as KDD Cup 1999 [6]. Nevertheless, these approaches depended on manual feature engineering, limiting their adaptability and scalability [7].

Deep learning has emerged as the state-of-the-art paradigm for IDS, offering automatic hierarchical feature extraction from raw data. Vinayakumar et al. [8] demonstrated that LSTM networks achieve superior performance on sequential network traffic data. Lansky et al. [9] conducted a systematic review of deep learning IDS and concluded that transformer-based and hybrid models outperform traditional neural networks on high-dimensional authentication logs. CNN-based models have also been applied to IDS, effectively exploiting spatial correlations in structured login records [10].

Despite these advances, open challenges persist: the need for large, balanced, and representative training datasets; high computational overhead for real-time inference; the interpretability deficit inherent in deep architectures (the 'black-box' problem); and the vulnerability of trained models to adversarial perturbations [11]. The proposed system addresses several of these challenges through architectural design choices described in the following sections.

Table I. Comparative Summary of IDS Approaches

Approach	Detection Type	Adaptability	Real-Time	Accuracy
Signature-based	Known attacks only	Low	Yes	Moderate
Rule-based (threshold)	Known patterns	Low	Yes	Low–Moderate
ML (SVM/DT/KNN)	Known & some unknown	Moderate	Partial	Moderate–High
Deep Learning (ANN)	Known & unknown	High	Yes	High
Deep Learning (LSTM)	Sequential/temporal	High	Yes	Very High
Proposed System (ANN+LSTM)	Multi-vector login attacks	Very High	Yes	Very High

III. SYSTEM ARCHITECTURE

The proposed IDS adopts a modular, layered architecture comprising six principal components: (1) User Interface Module, (2) Authentication Module, (3) Data Collection and Pre-processing Module, (4) Deep Learning Detection Engine, (5) Database Management Module, and (6) Alert and Response Module. Figure 1 illustrates the high-level system architecture and inter-module communication flows.

A. User Interface Module

The frontend is implemented using HTML5, CSS3, and JavaScript. It exposes a responsive login form through which users submit credentials, and a secured administrative dashboard for monitoring detected events, reviewing login logs, and configuring system parameters. All user inputs are transmitted to the backend via HTTPS-secured REST API calls to prevent interception.

B. Authentication and Data Collection Module

Upon receiving a login request, the backend collects a comprehensive set of contextual metadata alongside the submitted credentials. These include the source IP address, geolocation (resolved via MaxMind GeoIP), User-Agent string (device and browser type), timestamp, historical login frequency for the source IP, and the success/failure status of previous attempts. All data points are persisted to the database and simultaneously forwarded to the pre-processing pipeline for immediate analysis.

C. Pre-processing Module

Raw login records require transformation before deep learning inference. The pre-processing pipeline performs: (i) missing-value imputation using column means for numerical features and 'Unknown' for categorical ones; (ii) label encoding and one- hot encoding of categorical variables such as device type and country code; (iii) Min–Max normalisation of numerical features (login count, failed-attempt ratio, inter-arrival time) to the [0, 1] range; and (iv) feature selection retaining the twelve highest-information-gain attributes identified during offline training.

D. Deep Learning Detection Engine

The detection engine hosts two complementary models trained offline and served via a Flask-based inference API. An ANN model handles stateless, feature-vector classification, while an LSTM model analyses fixed-length sequences of recent login events (window size = 10) to capture temporal attack patterns. Each model outputs a probability score $\in [0, 1]$; scores above a configurable threshold τ (default 0.5) trigger the alert pipeline. The ensemble score is computed as the weighted average of both model outputs.

E. Database Module

A MySQL relational database stores three primary tables: Users (hashed credentials, account metadata), Login_Logs (complete per-attempt feature vectors), and Attack_Logs (confirmed or suspected intrusion events with classification labels). MongoDB is used as an auxiliary document store for unstructured event metadata. Periodic snapshots of Login_Logs are exported for model re-training cycles.

F. Alert and Response Module

When the detection engine flags an attempt, the response module executes a tiered action policy: (i) risk score 0.5–0.7 triggers a CAPTCHA challenge; (ii) 0.7–0.9 enforces OTP multi-factor authentication;
(iii) above 0.9 temporarily blocks the source IP and dispatches an email alert to the registered administrator account. All actions and their outcomes are logged for forensic analysis.

IV. DEEP LEARNING METHODOLOGY

A. Artificial Neural Network (ANN) Model

The ANN is a fully connected feedforward network trained for binary classification (benign = 0, attack
= 1). The architecture comprises an input layer sized to the feature vector dimension (twelve features), three hidden layers with 128, 64, and 32 neurons respectively, each followed by batch normalisation and a ReLU activation, and a single-neuron sigmoid output layer. Binary cross-entropy is used as the loss function, optimised using the Adam

optimiser (learning rate = 0.001). Dropout (rate = 0.3) is applied after each hidden layer to mitigate overfitting.

B. Long Short-Term Memory (LSTM) Model

The LSTM model processes a temporal sequence of the ten most recent login events from a given source, enabling the detection of time-distributed attack patterns — for instance, a brute force campaign that deliberately spaces attempts to evade rate-limiting controls. The network consists of two stacked LSTM layers (64 units each) with recurrent dropout (rate = 0.2), followed by a dense layer (32 neurons, ReLU) and a sigmoid output. The model is trained with sequences labelled at the sequence level (benign/attack) using teacher forcing.

C. Training Protocol

Both models are trained on a curated dataset combining the NSL-KDD benchmark dataset (adapted to login features) with synthetic records generated by simulating brute force and credential stuffing campaigns against a test application. The combined dataset contains approximately 120,000 records, partitioned 80:20 for training and testing. Class imbalance (typical login datasets contain fewer attack samples) is mitigated using SMOTE oversampling on the minority class. Both models are trained for
100 epochs with early stopping (patience = 10) monitored on validation loss. Hyperparameter optimisation is performed using Bayesian search.

Table II. Deep Learning Model Hyperparameters

Parameter	**ANN**	**LSTM**
Input size	12 features	10 × 12 (seq × feat)
Hidden layers	3 (128-64-32)	2 LSTM (64-64) + Dense (32)
Activation	ReLU + Sigmoid	Tanh/Sigmoid + ReLU + Sigmoid
Optimiser	Adam (lr = 0.001)	Adam (lr = 0.001)
Loss function	Binary Cross-Entropy	Binary Cross-Entropy
Dropout	0.3	0.2 (recurrent)
Batch size	64	32
Epochs	100 (early stopping)	100 (early stopping)

V. SYSTEM IMPLEMENTATION

The system is implemented using Python 3.10 as the primary language. TensorFlow 2.12 and Keras are used for model definition and training. The backend REST API is built with Flask 3.0 and exposes endpoints for login event submission, model inference, and administrative reporting. The frontend employs vanilla JavaScript with Bootstrap 5 for responsive layout. MySQL 8.0 serves as the primary relational store, and MongoDB 7.0 handles document-structured event metadata.

Model serialisation uses the TensorFlow SavedModel format. The inference endpoint deserialises incoming login feature vectors, applies the saved normalisation scaler, and returns a JSON payload containing the predicted class, ensemble probability score, and recommended response action. Average inference latency is measured at 45 ms per request on a standard cloud compute instance (Intel Xeon, 4 vCPUs, 16 GB RAM), well within the sub-second threshold required for transparent user experience.

Containerisation via Docker ensures environment reproducibility across development, staging, and production deployments. A CI/CD pipeline implemented in GitHub Actions automates testing and model re-deployment upon new training runs.

Table III. Technology Stack

Layer	Technology	Purpose
Frontend	HTML5, CSS3, Bootstrap 5, JavaScript	Login interface & admin dashboard
Backend	Python 3.10, Flask 3.0	API routing, business logic
ML Framework	TensorFlow 2.12 / Keras	Model training & inference
Data Processing	NumPy, Pandas, Scikit-learn	Feature engineering, normalisation
Primary Database	MySQL 8.0	Users, login logs, attack logs
Auxiliary Store	MongoDB 7.0	Unstructured event metadata
Containerisation	Docker 24	Reproducible deployment
Version Control	Git / GitHub	Source & model management

VI. RESULTS AND DISCUSSION

The proposed system is evaluated on a held-out test set comprising 24,000 login records (80% benign, 20% attack). Evaluation metrics include accuracy, precision, recall, F1-score, and Area Under the ROC Curve (AUC-ROC). Baseline comparisons are made against (i) a rule-based threshold system (block after 5 consecutive failures), (ii) SVM with RBF kernel, and (iii) an isolated ANN trained without the LSTM ensemble component.

A. Classification Performance

Table IV presents the comparative classification performance. The proposed ensemble (ANN + LSTM) achieves an accuracy of 97.6%, precision of 96.8%, recall of 98.1%, F1-score of 97.4%, and AUC-ROC of 0.991. The LSTM's ability to model temporal login sequences is particularly beneficial for detecting slow-rate brute force attacks that deliberately space attempts to evade static lockout policies, contributing a 2.3% improvement in recall over the standalone ANN.Table IV. Comparative Classification Performance

Method	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	AUC-ROC
Rule-based threshold	78.3	71.2	68.4	69.8	0.741
SVM (RBF kernel)	88.7	86.3	85.9	86.1	0.912
ANN (standalone)	94.2	93.1	95.8	94.4	0.968
Proposed: ANN + LSTM	97.6	96.8	98.1	97.4	0.991

B. Attack-Type Specific Detection

The system is tested against four distinct attack categories simulated in the evaluation environment. Brute force attacks (rapid consecutive failed attempts from a single IP) are detected with 99.1% recall, as the LSTM rapidly identifies the escalating failure-rate signature. Credential stuffing attacks (valid-appearing credentials from distributed IPs) achieve 96.3% recall, with geolocation inconsistency features contributing most to classification. Dictionary attacks achieve 97.8% recall. Slow-rate attacks (one attempt per hour from rotating IPs) achieve 93.4% recall — the most challenging scenario — where LSTM's long-sequence memory provides the critical advantage.

C. False Positive Analysis

The system records a false positive rate (FPR) of 1.8%, meaning legitimate users are incorrectly challenged on approximately 1 in 56 sessions. This is primarily attributable to users logging in from new geographic regions or switching devices without prior history. Personalised behavioural baselines, built over the first 30 successful logins per user, reduce the FPR to below 0.9% after the baseline establishment period. This compares favourably to the 12.4% FPR observed in the rule-based baseline.

D. Real-Time Performance

Inference latency is measured across 10,000 requests on the deployment server. The ANN inference path averages 18 ms; the LSTM path, which requires sequence retrieval and matrix operations, averages 45 ms. The ensemble latency of 45 ms (parallelised execution) is imperceptible to end users and imposes no meaningful overhead on login throughput. Under a simulated load of 500 concurrent login requests per second, the system maintains consistent classification latency with a 95th-percentile response time of 68 ms.

E. Discussion

The experimental results confirm that deep learning — particularly the ANN + LSTM ensemble — substantially outperforms conventional approaches across all evaluation metrics. The LSTM component's ability to model sequential login behaviour is the key differentiator for detecting sophisticated, temporally distributed attacks. The modular architecture ensures that individual components (pre-processing, models, response rules) can be updated independently as the threat landscape evolves. A noted limitation is the computational cost of periodic model retraining, which currently requires approximately 3.2 GPU-hours for the full dataset; future work will investigate online learning to reduce this burden.

VII. CONCLUSION AND FUTURE WORK

This paper has presented a Web-Based Intrusion Detection System for login attack detection, combining ANN and LSTM deep learning models within a scalable, modular web architecture. The system achieves 97.6% accuracy and 0.991 AUC-ROC on a balanced evaluation dataset, significantly surpassing rule-based and classical machine learning baselines. Real-time detection latency of 45 ms and low false positive rates confirm the system's suitability for production deployment in latency-sensitive applications such as online banking and enterprise portals.

Key contributions of this work include: (i) a comprehensive multidimensional feature set for login event characterisation; (ii) an ANN + LSTM ensemble strategy that exploits both instantaneous and temporal behavioural signals; (iii) a tiered, risk-proportional response mechanism; and (iv) a Docker-containerised, CI/CD-enabled deployment pipeline ensuring reproducibility.

Future research directions include: (a) integration of Transformer and attention-based architectures for improved long-range temporal dependency modelling; (b) Explainable AI (XAI) techniques — specifically SHAP values — to enhance decision transparency for security analysts; (c) federated learning to enable cross-organisational model training without sharing sensitive login data; (d) adversarial training to harden models against evasion attacks; and (e) cloud-native, auto-scaling deployment on AWS Lambda or Google Cloud Run for globally distributed applications.

Acknowledgement

The authors express sincere gratitude to the Department of Computer Science and Information Technology, Vels Institute of Science, Technology and Advanced Studies (VISTAS), Chennai, for providing the computational resources and research environment that supported this work. Special thanks to Dr. R. Parameswari, Professor and Head of the Department, for her continued encouragement.

References

[1] A. Khraisat, I. Gondal, and P. Vamplew, 'A Comprehensive Survey on Intrusion Detection Systems,' Cybersecurity, Springer, 2019.

[2] Y. Xin, L. Kong, and Z. Liu, 'Machine Learning and Deep Learning Methods for Cybersecurity,' IEEE Access, vol. 6, pp. 35365–35381, 2018.

[3] R. Vinayakumar et al., 'Deep Learning Approach for Intelligent IDS,' Journal of Information Security and Applications, 2020.

[4] D. E. Denning, 'An Intrusion-Detection Model,' IEEE Trans. Software Eng., vol. SE-13, no. 2, pp. 222–232, 1987.

[5] S. Axelsson, 'Intrusion Detection Systems: A Survey and Taxonomy,' Tech. Report, Chalmers Univ., 2000.

[6] A. L. Buczak and E. Guven, 'A Survey of Data Mining and Machine Learning Methods for Cybersecurity Intrusion Detection,' IEEE Communications Surveys & Tutorials, vol. 18, no. 2, pp. 1153–1176, 2016.

[7] V. Chandola, A. Banerjee, and V. Kumar, 'Anomaly Detection: A Survey,' ACM Computing Surveys, vol. 41, no. 3, 2009.

[8] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, and S. Venkatraman, 'Robust Intelligent Malware Detection Using Deep Learning,' IEEE Access, 2019.

[9] J. Lansky et al., 'Deep Learning-Based Intrusion Detection Systems: A Systematic Review,' IEEE Access, vol. 9, pp. 101574–101599, 2021.

[10] M. Soltani et al., 'A Survey of Network-Based IDS Using Deep Learning,' J. Network and Computer Applications, 2021.

[11] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016.

[12] T. Ahmad et al., 'Network Intrusion Detection Using Deep Learning,' Computational Intelligence and Neuroscience, 2022.

[13] N. Moustafa and J. Slay, 'UNSW-NB15: A Comprehensive Data Set for Network IDS,' Proc. MilCIS, pp. 1–6, 2015.

[14] A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd ed. O'Reilly Media, 2019.

[15] TensorFlow Documentation, Google AI, https://www.tensorflow.org, 2024