Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 17, No. 1, 2026

Review on Portal Intelligent Authentication Algorithm Based on Artificial Intelligence

Li Wang

School of Chengdu Jincheng University, Chengdu, Sichuan, 61000, China

Abstract: The Portal authentication system constitutes a core component of modern network authentication systems. The vast majority of conventional authentication schemes rely on static username-password authentication mechanisms, which inherently suffer from security vulnerabilities. Furthermore, mandatory periodic password updates and high password complexity requirements degrade user experience significantly. This paper delivers a comprehensive and integrated analysis of state-of-the-art intelligent authentication algorithms for network access authentication, and thoroughly dissects three technical implementation frameworks: user behavior modeling, anomalous access behavior detection, and authentication decision optimization. The research focuses on elaborating the fundamental principles and practical deployment details within real authentication systems for temporal behavior modeling based on Long Short-Term Memory (LSTM), integrated anomaly detection architecture, and decision optimization strategies. It systematically organizes the algorithm execution workflow and simulation testing schemes. Ultimately, this paper discusses and analyzes the prevailing technical bottlenecks and prospective development trends of high-performance intelligent authentication algorithms.

Keywords: Portal Authentication; User Behavior Modeling; Anomaly Detection; Reinforcement Learning; Intelligent Authentication.

1. Introduction

Portal authentication technology is widely adopted on campus networks, corporate internal networks and public Wi-Fi hotspots. When users first connect to the network, their web browsers will redirect to a dedicated authentication page. Users must submit valid account credentials to pass identity verification before they can gain network access. This conventional authentication method is straightforward to operate, yet it comes with notable security risks in real-world deployment. Systems built purely around static passwords are prone to data leakage. They cannot flag suspicious login activities, nor can they support fine-grained risk assessment for network access scenarios[1].

Advancements and iterative upgrades in network security and access control technology over recent years have laid robust technical support for feature upgrades and performance tuning of Portal authentication platforms. Authentication platforms can extract and analyze behavioral traits generated during user access, build standardized and precise user behavior profiles, spot high-risk and unauthorized login attempts, and tweak authentication policies dynamically based on real-time access risk levels. This paper carries out an in-depth analysis of core algorithms powering intelligent Portal authentication systems, covering three major research branches: user behavior modeling, anomaly detection algorithms, and authentication decision optimization[2].

2. User Behavior Modeling Algorithm

2.1. Temporal Behavior Feature Extraction

User login activities carry distinct temporal characteristics, such as login time intervals, sequences of visited pages, and device switching frequency. Such indicators play a vital role in user identity recognition and anomalous access behavior detection.

2.1.1. LSTM Network Structure

Long Short-Term Memory (LSTM) networks address the vanishing gradient problem of traditional RNN through gating mechanisms and perform excellently in temporal data modeling. The core of LSTM consists of three gating units: forget gate, input gate, and output gate [3].

Forget gate: $\mathrm{f}_t = \sigma(W_t \cdot [h_{t-1}, x_t] + b_t)$

Input gate: $\mathrm{i}_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$

Candidate cell state: $\overline{\mathrm{C}}_t = \tanh(W_{\mathrm{c}} \cdot [h_{t-1}, x_t] + b_c)$

Updated cell state: $\mathrm{C}_t = \mathrm{f}_t \odot \mathrm{C}_{\mathrm{t}} + i_t \odot \overline{\mathrm{C}}_{\mathrm{t}}$

Output Gate: $\mathrm{o}_t = \sigma(W_{\mathrm{o}} \cdot [h_{t-1}, x_t] + b_o)$

Hidden output state: $\mathrm{h}_t = o_t \odot \tanh(\mathrm{C}_{\mathrm{t}})$

where σ stands for the sigmoid activation function, and $\odot$ denotes the Hadamard product (element-wise multiplication).

2.1.2. LSTM Model Implementation

The implementation of the LSTM user behavior modeling model includes the following key steps: First, the model initialization phase requires setting parameters such as input dimension, hidden layer dimension, number of network layers, and dropout ratio. The input dimension corresponds to the number of extracted user behavior features, the hidden layer dimension is typically set to 128 or 256, the number of network layers is generally chosen between 2 to 3, and the dropout ratio is used to prevent overfitting.

The LSTM layer configuration adopts a bidirectional structure with the batch_first parameter set to True to accommodate the input data dimension format. For multi-layer networks, dropout is applied only between layers. The multi-head attention layer uses 8 attention heads, which can effectively capture key information in temporal data.

The forward propagation process sequentially includes: first extracting temporal features through the LSTM layer, outputting hidden states and cell states; then inputting the

LSTM output into the multi-head attention layer to obtain weighted feature representations; then performing global average pooling on the attention output to compress into a fixed-dimensional vector; and finally classifying through two fully connected layers using ReLU activation function and dropout, with the final output using the sigmoid function to map results to the 0-1 interval[4].

2.1.3. Feature Extraction Process

The user behavior feature extraction process is shown in Fig.1.

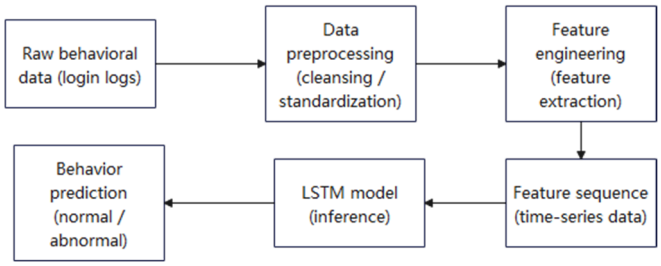

Fig 1. Flowchart of User Behavior Feature Extraction and Modeling

2.2. Behavioral Feature Engineering

2.2.1. Temporal Features

Temporal features reflect the time patterns of user behavior, mainly including: 1) Login time interval: the time difference between consecutive logins; 2) Login time period distribution: distribution of login counts within 24 hours of a day; 3) Periodic features: differences in login patterns between weekdays and weekends; 4) Login frequency: number of logins per unit time.

2.2.2. Behavioral Features

Behavioral features reflect user interaction patterns: 1) Mouse features: average movement speed (pixels/ millisecond), movement direction change rate, click interval time; 2) Keyboard features: average typing speed (characters/second), key press duration, key interval time distribution; 3) Page interaction features: page dwell time, scrolling behavior patterns, form filling duration.

2.2.3. Contextual Features

Contextual features provide environmental information support for authentication scenarios, which are mainly divided into three dimensions. First, Network features: IP address, network type such as WiFi and 4G. Second, Device features: device fingerprint, operating system, browser type. Third, Geographic location: IP geolocation, GPS coordinates (mobile devices).

2.2.4. Feature Extraction Implementation

Three core modules are adopted to construct the feature extractor for user behavior, namely temporal feature extraction, behavioral feature extraction and feature normalization. For temporal feature extraction, the system first collects all timestamps from login sequences, calculates the time interval between every two adjacent timestamps and converts these values into hours. Afterwards, it computes statistical metrics including the mean, standard deviation, median, minimum and maximum of all intervals, as well as the total number of login records. To describe the distribution of login hours, each timestamp is mapped to an integer ranging from 0 to 23. A histogram is applied to count login volumes within each hour slot, followed by normalization to output a 24-dimensional feature vector that reflects time distribution patterns.

Behavioral feature extraction separately processes mouse and keyboard operation events. When handling mouse data, coordinate points of all mouse actions are extracted at first. The Euclidean distance and time difference between consecutive coordinate points are then calculated. Dividing distance by time difference yields mouse moving speed, based on which the average speed, standard deviation of speed and total moving distance can be further derived. If no mouse events exist, related features are set to 0. For keyboard events, it separates key press and release events, calculates key press duration by matching press and release events of the same key, extracts press event timestamps to calculate intervals between consecutive key presses, and finally computes mean and standard deviation of press durations and intervals. If no keyboard events exist, related features are set to 0.

The feature standardization method uses StandardScaler to perform Z-score standardization on feature vectors, ensuring all features have zero mean and unit variance to facilitate subsequent model training.

3. Anomaly Detection Algorithms

3.1. Isolation Forest Algorithm

Isolation Forest is an unsupervised anomaly detection algorithm that isolates data points by randomly partitioning the feature space. Anomalous points, being fewer in number and having different feature values from normal data, are more easily isolated.

3.1.1. Algorithm Principle

Isolation Forest constructs multiple isolation trees. The construction process of each tree is as follows: 1) Randomly select a feature dimension; 2) Randomly select a split point within the value range of that dimension; 3) Recursively partition left and right subsets until stopping conditions are met (maximum depth reached or subset contains only one sample). Anomalous points have shorter average path lengths, while normal points have longer average path lengths.

3.1.2. Algorithm Implementation

The anomaly detection module runs through three core steps, including system initialization, parameter calibration and behavior judgment.

Two basic parameters need to be configured during system initialization, which are the preset proportion of suspected abnormal samples and fixed random seed. Such settings can guarantee stable test outcomes and make all detection results repeatable. We set the core operating parameters of the detector as follows: one hundred separate isolation trees are built, the maximum sampling size is obtained through self-adaptive methods, all feature dimensions take part in computation, bootstrap resampling is turned off, and full-CPU parallel computing is enabled to raise the overall detection speed.

At the parameter calibration phase, raw input data will be standardized into unified numerical arrays first. Special adaptive transformation is carried out on one-dimensional data to unify data formats. After data preprocessing, the detector finishes parameter calibration and formulates judgment rules with the collected data set. Once calibration is finished, we compute the decision scores of all samples within the training set and extract their mean value and standard deviation. These statistical values lay the groundwork for self-adaptive adjustment of dynamic judgment thresholds.

When entering the behavior judgment phase, the system will check the calibration state of the detector first to

guarantee normal running. Next, input data will go through standardization and adaptive processing for one-dimensional structural features. The built-in calculation rules are used to calculate risk scores for each sample. Lower raw scores mean a higher chance of abnormal network access behaviors. All original scores are normalized to the interval from 0 to 1; larger normalized values represent more obvious abnormal behavior features. Instead of manually setting fixed thresholds, this system uses a dynamic threshold solving method to divide judgment boundaries. In the end, the classification results of normal and abnormal behaviors are exported according to the confirmed threshold standards[5].

The dynamic threshold calculation method is based on statistics of training set scores, using the formula mean + k × std, where k defaults to 2.0. After calculation, the threshold is normalized to the 0-1 range for comparison with normalized anomaly scores[5].

3.2. Integrated Anomaly Detection Framework

Single detector has inherent performance defects; ensemble learning improves detection accuracy and robustness by combining multiple complementary algorithms.

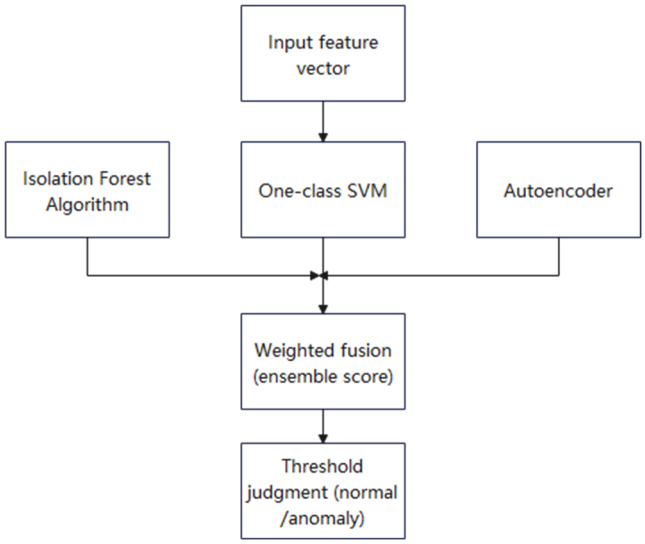

Fig 2. Architecture of Ensemble Anomaly Detection Framework

The implementation of the integrated anomaly detector includes four main parts: initialization, model construction, training, and prediction.

The initialization phase sets the anomaly sample ratio while initializing three detectors: Isolation Forest using 100 independent isolation tree structures, One-Class SVM using RBF kernel function, and Autoencoder to be constructed. Additionally, it initializes a data standardizer and weight configuration, with algorithm weights of 0.4, 0.3, and 0.3 respectively[6].

The autoencoder construction method creates an encoder-decoder structure. The encoder progressively compresses the input dimension to 64, 32, and 16 dimensions, while the decoder progressively decompresses back to the original dimension. The model uses Adam optimizer, mean squared error loss function, and mean absolute error as the evaluation metric.

The training method first standardizes the input data. It then trains the three detectors separately: Isolation Forest and One-Class SVM are trained directly using standardized data; the autoencoder uses early stopping to prevent overfitting, monitoring validation loss with patience set to 10, training for 100 epochs with batch size of 32, and validation set ratio of 20%.

The prediction method first checks if the model has been trained, then standardizes the input data. It obtains decision scores from the three models: Isolation Forest and One-Class SVM use normalization methods, while the autoencoder scores are obtained by calculating reconstruction errors and normalizing them. Finally, weighted fusion is used to obtain the integrated score, with threshold set to 0.5, generating prediction labels based on the threshold. The score normalization method linearly maps scores to the 0-1 range using the formula (max - score) / (max - min), where smaller scores indicate higher likelihood of anomaly.

The autoencoder scoring method first uses the trained model to reconstruct input data and calculates the mean squared error between reconstructed and original data. The error is then normalized to the 0-1 range to obtain the anomaly score.

3.3. Real-time Anomaly Detection Process

Real-time anomaly detection needs to meet low-latency and high-concurrency requirements. The processing flow is shown in Fig.3.

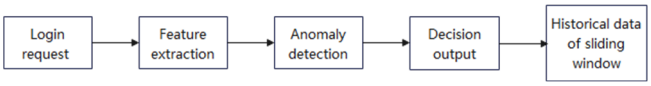

Fig 3. Real-time Anomaly Detection Flow

4. Authentication Decision Optimization Algorithms

4.1. Reinforcement Learning Framework

Reinforcement learning learns optimal policies through agent-environment interaction. In the Portal authentication scenario, the agent selects authentication actions based on user behavior states and adjusts policies based on feedback from security and user experience.

The code implementation of the Q-Learning agent consists of six core functions: initialization, state discretization, action selection, learning iteration, experience storage and experience replay.

We configure all hyperparameters in the initialization process, including learning rate, discount factor, exploration rate and exploration rate decay. The learning rate adjusts the step size for Q-value updates. The discount factor quantifies the weight assigned to future rewards, while the exploration rate sets the probability that the agent executes random exploratory actions. A defaultdict data structure is adopted to build the Q-table. Its keys correspond to system states, and values store the mapping between each action and its matched Q-value. The experience replay buffer is realized with a deque container, and the maximum storage capacity of this buffer is limited to 10,000 samples.

The state discretization function transforms continuous state variables into discrete tuples. Risk scores are multiplied by 10 and rounded to generate 11 discrete risk levels ranging from 0 to 10. Hour values are divided by 2 before rounding, splitting the 24-hour day into 12 separate time slots. Device IDs and location IDs are converted into integer values of either 0 or 1. This function finally outputs a four-element tuple as the standardized discrete state.

The action selection component follows the ε-greedy

strategy. During model training, if a generated random number is smaller than the current exploration rate, the agent picks an arbitrary action for exploration. Otherwise, it selects the action with the highest Q-value for exploitation. When multiple actions share the identical maximum Q-value, one candidate will be chosen at random.

The learning method implements the core update rule of Q-Learning. First, the current state and next state are discretized, then the Q-value of the current action and the maximum Q-value of all actions in the next state are obtained from the Q-table. According to the Bellman equation, the Q-value is updated: the new Q-value equals the current Q-value plus the learning rate multiplied by (immediate reward plus discounted maximum future Q-value minus current Q-value). Finally, the exploration rate is gradually decreased according to the exploration rate decay coefficient until reaching the minimum value of 0.01[7].

The experience storage method adds tuples composed of state, action, reward, next state, and done flag to the experience replay buffer. The experience replay method randomly samples a batch of experiences from the experience buffer and sequentially calls the learning method to update the Q-table. When the number of experiences in the buffer is less than the batch size, it returns directly; otherwise, it randomly samples the specified number of experience indices.

4.2. Authentication Decision Process

The reinforcement learning-based authentication decision process is shown in Fig.4.

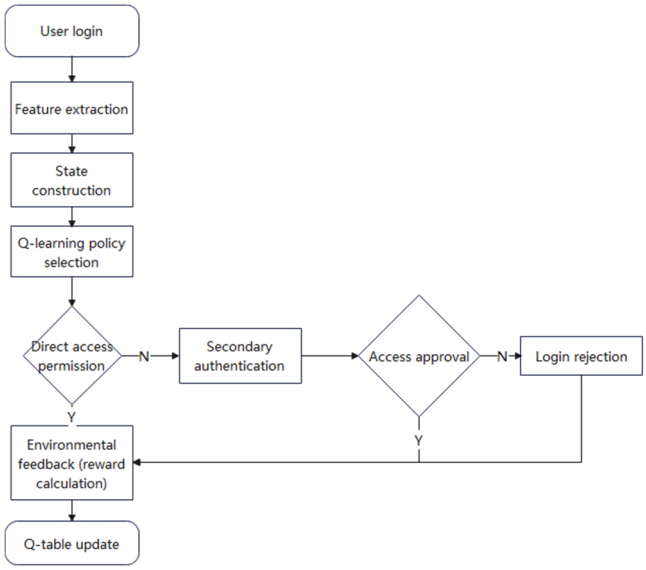

Fig 4. Reinforcement Learning-based Authentication Decision Flow

5. Simulation Experiment

5.1. Dataset Generation

To validate algorithm effectiveness, a simulation dataset needs to be constructed. The dataset contains two types of samples: normal user behavior and anomalous behavior. The design of the simulation data generator includes multiple methods: initialization, user profile generation, normal behavior generation, anomalous behavior generation, and mouse-keyboard event generation. The initialization method sets the number of users and simulation days, defaulting to 100 users and 30 days respectively.

The user profile generation method creates a profile for each user containing user ID, device list, location list, common login time periods, and average login interval. The device list contains two common devices, the location list contains one common location, common login time periods randomly select 5 time points between 8 AM and 6 PM, and the average login interval follows a normal distribution with mean of 4 hours and standard deviation of 1 hour.

The normal behavior generation method generates a certain number of normal login records for a specified user. The reference time is set to the current time minus the simulation days. For each sample, a common login time period, day, and minute are randomly selected to generate the login time, a common device is randomly selected, and a random number of mouse and keyboard events are generated. Mouse event count ranges from 5 to 20, and keyboard event count ranges from 10 to 50. Each record is labeled as normal (label 0).

The anomalous behavior generation method generates anomalous login records. Login time is selected from anomalous time periods (0-3 AM or 8-11 PM), using an anomalous device (non-common device ID), and using an anomalous location (random location). Mouse and keyboard event speeds are multiplied by anomaly multipliers (2.5 and 2.0 respectively) to simulate anomalous interaction behavior. Each record is labeled as anomalous (label 1).

The mouse event generation method starts from a random starting point, with movement time intervals following an exponential distribution, movement direction following a normal distribution, and coordinates limited to screen range (1920×1080). The keyboard event generation method similarly uses exponential distribution to generate key press and release time intervals, randomly selecting alphanumeric characters. The dataset generation method generates normal samples and anomalous samples for each user, merges them, randomly shuffles the order, and converts to DataFrame format for return.

5.2. Evaluation Metrics

We adopt several standard metrics to evaluate the performance of the anomaly detection algorithm.

Four core indicators are calculated via the evaluation function, namely accuracy, precision, recall and F1-score. Accuracy stands for the overall correctness of classification results. Precision refers to the proportion of real abnormal samples among all data points predicted as anomalies. Recall measures how many actual abnormal samples can be successfully identified. The F1-score is calculated as the harmonic mean of precision and recall.

A confusion matrix categorizes all prediction outcomes into four groups: true negatives, false positives, false negatives and true positives. False alarm rate and missing alarm rate can be further derived from the matrix. The false alarm rate corresponds to the percentage of normal samples misjudged as abnormal, whereas the missing alarm rate represents the share of abnormal samples incorrectly labeled as normal. When division by zero occurs during calculation, the program returns zero values to eliminate division-by-zero runtime errors.

5.3. Experimental Process

The complete experimental evaluation process is shown in Fig.5.

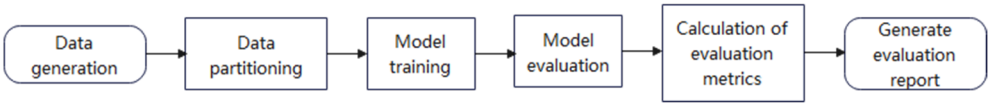

Fig 5. Experiment Evaluation Process

5.4. Experimental Example

The whole experimental workflow is split into six successive steps: data generation, feature extraction, dataset segmentation, detector calibration, behavior judgment and performance measurement.

In the data generation step, a simulation data generator is used to build a 30-day user behavior dataset covering 100 separate users. We summarize and record basic statistical information of this dataset, such as total sample quantity, the count of normal samples and anomalous samples.

A standard feature extraction framework is adopted in the feature extraction phase to extract temporal features and interactive behavior features from each user access log. Temporal features mainly cover statistical indicators of login intervals and time distribution features of login activities. Interactive behavior features are derived from mouse and keyboard operation records. All extracted feature dimensions are combined to form a standardized data frame, and the matching label vector is generated for subsequent experimental analysis.

We split the full dataset into training and test subsets via stratified sampling, where test data takes up 30 percent of the total volume. Fixed random parameters are set to guarantee repeatable experimental results. Stratified segmentation keeps the label distribution consistent across two subsets and prevents deviation in sample distribution.

During detector calibration, we build an integrated anomaly detection framework with the preset abnormal sample ratio set to 10%. Training and optimization of detection parameters are carried out on the finished training dataset to finish parameter configuration for the whole detection system.

The calibrated detection system processes the test dataset in the judgment stage, and outputs classification labels and numerical anomaly scores for every test sample.

In the performance measurement phase, quantitative evaluation functions are invoked to calculate multiple detection metrics and output complete experimental outcomes. All floating-point evaluation values are rounded to four decimal places, while integer statistical indicators are kept as original values to guarantee data precision and consistent results.

6. Conclusion

This paper sorts out various intelligent Portal authentication algorithms powered by artificial intelligence, and elaborates on three core technical modules, namely user behavior modeling, anomaly detection and authentication decision optimization. We adopt long short-term memory networks to extract and model time-series behavior features of users. An ensemble learning framework is built to combine three anomaly detection methods, including Isolation Forest, One-Class SVM and Autoencoder. Moreover, an adaptive optimization mechanism for authentication decisions is designed with reinforcement learning techniques.

Simulation tests reveal that the presented ensemble anomaly detection framework achieves better detection performance than any single detection algorithm. The decision-making system built on reinforcement learning can dynamically modify authentication policies according to real-time running status. It balances security requirements and user experience simultaneously.

Three promising research paths can be explored in follow-up work. First, develop lightweight feature processing structures suitable for deployment on edge devices. Second, study privacy-preserving distributed computing solutions dedicated to network access authentication scenarios. Third, construct an authentication architecture with multi-dimensional feature fusion, so as to raise identification accuracy and strengthen overall system robustness.

References

[1] Hu, Y. T. (2024). Security defense of campus network portal access authentication. *Computer Programming Skills & Maintenance*, 4, 170–173.

[2] Zhao, Z. (2020). Research on security authentication architecture and methods of wireless city. *Modern Information Technology*, 4(12), 151–153.

[3] Zou, F. (2025). Research on network intrusion detection method based on improved CNN-LSTM model. *Sci-Tech Innovation and Productivity*, 46(10), 117–120.

[4] Zhen, H. Y., & Wan, Z. Y. (2025). Abnormal detection and repair of broadband network based on neural network. *China Broadband*, 21(8), 22–24.

[5] Xu, W., & Leng, J. (2025). Research on large-scale network intrusion attack detection based on improved isolation forest. *Modern Electronic Technology*, 48(15), 98–102.

[6] Huang, Y. T., Pei, X. B., Kong, L. B., et al. (2019). Research on power outlier user detection algorithm based on ant colony algorithm optimized one-class SVM. *Automation & Instrumentation*, 5, 111–114.

[7] Liu, S., Dong, X. S., & Zhao, W. (2025). Research on agent path planning based on improved Q-learning algorithm. *Computer Era*, 11, 1–6.