Computer Science and Interdisciplinary Research Journal
Volume 2, Issue 2, 2025. ISSN: 3033-1218, DOI: 10.70862/CSIR.2025.0202-17

Integration of GenAI and Behavioral Analysis in an Authentication System

Georgi Atanasov, Kostadin Yotov, Emil Hadzhikolev

University of Plovdiv Paisii Hilendarski, 236, Bulgaria Blvd, 4027 Plovdiv, Bulgaria

stu2501322014@uni-plovdiv.bg, kostadin_yotov@uni-plovdiv.bg, hadjikolev@uni-plovdiv.bg

Abstract – This paper presents a model of a security-enhanced web authentication system that combines behavioral analysis, generative artificial intelligence, and an event-driven architecture. The aim of the study is to develop a dynamic monitoring and alerting mechanism capable of identifying unusual activity based on contextual and behavioral indicators, without imposing additional authentication steps on the user. The system leverages Redis with Pub/Sub and TTL notifications for reactive event processing, generative AI for automatic summarization of logs into a human-readable format, and the Isolation Forest algorithm for experimental detection of behavioral anomalies. The results demonstrate that the proposed architecture can effectively identify suspicious sessions, provide interpretable textual explanations of user actions, and react to key events in real time while maintaining low latency.

Keywords – authentication, generative artificial intelligence, GenAI, behavioral security, reactive architecture, anomaly detection.

1. Introduction

Over the past two decades, the internet ecosystem has experienced exponential growth – both in the number of active users and in the diversity and complexity of online services. The digitalization of business, education, and the public sector has led to a situation in which vast amounts of sensitive data – personal, financial, and professional – are transmitted, stored, and processed in networked environments [1-3]. In this context, authentication and authorization have become critical components of cybersecurity, determining not only the technical reliability of a system but also the level of trust placed in it by end users.

Traditional authentication methods, based solely on username–password combinations, can no longer address the dynamic threats of the modern digital landscape. Password leaks, phishing attacks, automated credential-stuffing bots, and the misuse of compromised accounts have become increasingly common [4-7]. According to the Verizon Data Breach Investigations Report [8], more than 70% of system breaches begin with unauthorized access obtained through compromised user credentials. This highlights the vulnerability of systems that rely on static authentication mechanisms.

Among the leading trends in authentication systems are multi-factor authentication (MFA), behavioral biometrics, and dynamic risk assessment, all of which provide continuous user validation [9-11]. Modern web applications no longer rely on a single initial login check; instead, they incorporate ongoing monitoring of user behavior, location, and access context.

An additional challenge stems from the increasing use of artificial intelligence (AI) and automated services acting on behalf of the user, which necessitates systems capable of distinguishing human actions from machine-generated ones. As a result, authentication is evolving from a static entry point into a dynamic trust mechanism that adapts its responses based on context, risk levels, and the history of interactions.

In this technological and social context, there arises a need for a new generation of authentication and access-control systems – ones that combine classical

DOI: https://doi.org/10.70862/CSIR.2025.0202-17

Corresponding author: Emil Hadzhikolev,
University of Plovdiv Paisii Hilendarski
Email: hadjikolev@uni-plovdiv.bg

Received: 30 November 2025.
Revised: 20 December 2025.
Accepted:21 December 2025.
Published: 21 December 2025.

BY-NC-ND © 2025 Georgi Atanasov, Kostadin Yotov, Emil Hadzhikolev. This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

The article is published with Open Access at http://csirjournal.com/

security principles with behavioral analysis, automation, and generative artificial intelligence (GenAI). Such systems not only validate the user but also understand the way the user interacts with the application, turning security into an intelligent and adaptive process.

The aim of the present study is to develop a dynamic and adaptive authentication and access-control system that enhances security trust through the integration of three key components:

- Behavioral security – monitoring request frequency, navigation paths, session duration, geographic location, and device type. These characteristics form a "digital behavioral fingerprint" that enables the detection of deviations typically associated with misuse or malicious activity.
- GenAI – automatic summarization of user sessions into clear, contextual reports that accelerate analysis and response.
- Intelligent alerting – automatic notifications triggered by anomalies, risks, or unusual changes in access patterns.

The study proposes a model of a dynamic and adaptive authentication system – Forest Gate. A prototype has been developed, with its source code publicly available at: https://github.com/georgi2005atanasov/forest_gate. Experiments conducted with this prototype show that the system operates reliably and stably in real-time across different user-behavior scenarios.

2. Technologies and Concepts for Authentication and Access Control

Authentication and access-control systems are among the oldest yet fastest-evolving components of modern cybersecurity. From simple username–password authentication to multifactor and context-aware approaches, developments in this field reflect the continuous pursuit of three key objectives: reliability, usability, and adaptability.

2.1. Classical and Modern Authentication Mechanisms

The classical authentication model based on a username and password has long been the dominant standard in web security [12-14]. However, its simplicity and ease of integration come with a significant drawback – the absence of adaptability to an evolving threat landscape. In an environment characterized by a growing number of cyberattacks, automated password-cracking tools, and social engineering techniques, static authentication methods are becoming increasingly ineffective.

A password represents a single factor of trust – knowledge that can be compromised through phishing, database leaks, or brute-force attacks. According to IBM Security [15], the average time to detect a breach involving stolen credentials exceeds 200 days, allowing attackers to operate unnoticed for extended periods.

Classical login mechanisms also suffer from a lack of context. The system does not distinguish whether the user is logging in from a familiar device, from an unusual geolocation, or with atypical behavioral patterns – for example, rapid navigation, high-volume API requests, or activity during abnormal hours. Therefore, even a successfully completed authentication does not guarantee that the access is legitimate [16-18].

Modern attacks such as credential stuffing, session hijacking, token replay, and keylogging exploit precisely this static nature. Since traditional systems do not track contextual factors, they cannot detect cases in which a valid token is used simultaneously from different countries – a scenario commonly referred to as impossible travel. In such situations, the absence of behavioral monitoring and dynamic risk assessment results in a delayed or entirely ineffective system response.

With the evolution of internet applications and web services requiring shared access across multiple clients and servers, new standards for secure identification have emerged. Among the most prominent are OAuth 2.0 and OpenID Connect, which provide mechanisms for delegated access through access tokens [19-21]. These standards allow users to authenticate themselves to different applications without revealing their passwords and enable centralized rights management. JSON Web Tokens (JWTs) have become the de facto standard for portable, self-contained tokens in distributed architectures and microservice environments [19,22].

Although these standards address important security challenges, they remain static in nature: authentication is performed only once, while access remains valid until the token expires. This makes them vulnerable to token theft, session reuse, and session hijacking attacks.

2.2. Behavioral Analysis and Contextual Authentication

Among contemporary trends, the use of behavioral analytics as a complement to classical authentication mechanisms stands out. This approach considers not only who is accessing the system but also how they do it – by analyzing the speed and sequence of actions, timing patterns, location, device type, navigation habits, and even typing behavior [9,10,23].

Through statistical models and machine-learning algorithms, the system can detect deviations from typical behavior and indicate potential risk.

Approaches such as continuous authentication and adaptive risk-based authentication enable real-time decision-making – for example, requesting an additional authentication factor or temporarily restricting access when suspicious activity is detected [11,24].

2.3. Use of AI in Security

In recent years, AI has become a key tool for analysis and automation in the security domain. Traditional machine-learning models – such as Random Forest, Support Vector Machines, and Neural Networks – are used for anomaly classification, phishing and spam detection, risk assessment, and attack prediction [25-27].

With the rise of GenAI, new possibilities have emerged – not only for threat detection but also for explainability and data interpretation. Generative models can transform technical logs and raw data into comprehensible textual descriptions that help administrators and researchers gain a faster overview of events [28-30]. This capability is particularly valuable when processing large volumes of data, where human analysis would be too slow or incomplete.

2.4. Anomaly Detection Algorithms: Isolation Forest

Anomaly detection algorithms are a core component of modern security systems. Among them, Isolation Forest (iForest) stands out for its efficiency on large datasets and its low computational complexity. Its principle is based on the idea that anomalies are "few and easier to isolate" – they require fewer partitions during the recursive data-splitting process [31-34].

2.5. Integrated Trust Systems

Recent research in the security domain increasingly focuses on integrated trust-driven architectures [35,36]. These systems combine technical, behavioral, and contextual data to construct a dynamic trust index for each user and each action. This concept forms the foundation of Zero Trust Security – a model in which access is granted not based on static permissions but according to the current risk assessment.

3. Development of Forest Gate

The Forest Gate system demonstrates an integrated security approach in which classical authentication mechanisms are enhanced through behavioral analysis and GenAI. The architecture is modular and distributed – each component is autonomous but communicates with the others through well-defined interfaces, ensuring transparency in data processing and flexibility for future extensions.

The architectural design is guided by three core requirements:

- Access and data security – protected communication between components using cryptographic keys and strictly limited information sharing.
- Traceability and explainability – detailed logs, session records, and summarized action reports that enable both technical and analytical inspection.
- Automation and adaptability – automatic collection of interaction data, AI-driven analysis via GenAI, and alerting in the presence of anomalies or risk events.

The technologies used in the implementation include Rust, PostgreSQL, Redis, NGINX, Docker, MaxMind GeoIP, SendGrid, and GenAI via OpenRouter. The anomaly detection algorithm employed is Isolation Forest.

3.1. Technologies Used

The implementation of the Forest Gate prototype is based on the integration of several complementary technologies that ensure high performance, stability, and system extensibility:

1. *Programming language and core logic – Rust.*

The system's core is implemented in Rust – a language that has become a standard for secure and efficient software development. Its ownership & borrowing model provides memory safety without a garbage collector and eliminates entire classes of vulnerabilities related to improper resource handling. Rust enables reliable implementation of asynchronous processes – request handling, session management, and Pub/Sub communication – without compromising security, which is critical when managing sensitive authentication data.

2. *Data management – PostgreSQL and Redis.*

PostgreSQL serves as the primary relational store for persistent data such as users, devices, login attempts, and configuration settings. Its ACID compliance and robustness make it suitable for critical systems. Redis functions as an in-memory cache and an inter-process communication mechanism. Thanks to Pub/Sub and keyspace notifications, the system can monitor user activity in real-time, react to session expiration, and trigger automated processes for analysis and alerting.

3. *Containerization and infrastructure – Docker and NGINX.*

The prototype is built using Docker Compose, which integrates the individual components into an isolated and reproducible environment. This greatly facilitates local testing and provides a foundation for

further deployment. NGINX operates as a reverse proxy responsible for request routing, caching, and TLS protection.

4. GenAI – OpenRouter.

GenAI models integrated via the OpenRouter API are used to analyze and summarize user behavior. They transform raw logs into concise and readable Markdown descriptions, improving the system's transparency and explainability. The prompts are designed to ensure that the model does not fabricate information but extracts only relevant facts – an essential property for internal auditing and security.

5. Anomaly detection – Isolation Forest.

Isolation Forest is used experimentally to identify atypical patterns in user behavior. The algorithm enables the detection of suspicious logins and anomalies in request frequency. Although not a central component, it demonstrates the potential to extend Forest Gate toward adaptive risk assessment, where the system responds based on the statistical abnormality of actions.

6. Intelligent alerting – SendGrid.

Integration with SendGrid enables automatic notifications triggered by anomalies, configuration changes, or other risk events. This functionality implements the principle of real-time monitoring – the system not only collects data but also actively communicates with administrators.

7. Geographical data – MaxMind GeoIP.

The system uses local GeoLite2-City, GeoLite2-ASN, and GeoLite2-Country databases to enrich requests with geographic information – country, city, and internet service provider (ASN/ISP). This enables the detection of scenarios such as impossible travel and sudden changes in network context.

3.2. Architecture

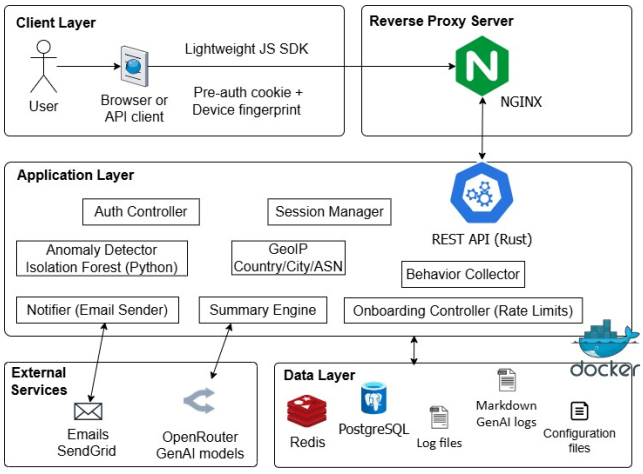


Figure 1. Architecture of Forest Gate.

The architecture of Forest Gate is multilayered and integrates several modules built on the technologies discussed earlier (Figure 1). The components communicate through well-defined interfaces and protocols, ensuring secure, transparent, and adaptive processing of user sessions. The system integrates classical authentication, behavioral analysis, intelligent alerting, and GenAI.

The client layer provides the interface between the user and the system and includes three main components:

- Web application or API client – used to initiate access requests (login, registration, profile updates, etc.).
- Pre-identification system – generates a temporary pre-auth cookie and a unique device fingerprint during the user's first visit. These markers allow the server to associate requests with a specific user and device without compromising anonymity.
- Lightweight JavaScript library – collects behavioral indicators (input timing, navigation speed, request frequency) and forwards them to the backend for analysis.

An NGINX reverse proxy server is positioned as an intermediate layer between the client and the business logic, responsible for request routing, caching, and TLS protection.

The business logic of the system is implemented in the application layer. The core components are developed in Rust, while the Isolation Forest algorithm is implemented in Python. The application server is responsible for authentication, data management and storage, event processing, and communication with the other modules in the architecture. The main modules in the server layer are:

- Onboarding Controller – used during initial registration or a user's first visit. It stores the device fingerprint and activates rate limits based on three parameters: IP address, visitorId (generated on first visit), and the pre-auth cookie.
- Auth Controller – validates credentials and issues sessions/tokens. Sessions can be cached in Redis and persisted when necessary.
- Behavior Collector – receives client-side events and stores them in Redis with a limited TTL. When a key expires, Redis triggers an expired event via Pub/Sub, which initiates subsequent processing.
- Summary Engine (GenAI) – a background service (flusher) that listens for Pub/Sub expiration notifications, collects accumulated events, and generates a concise Markdown summary of the behavior using OpenRouter.
- Anomaly/Risk Module – performs basic risk assessment and detects atypical access patterns using the Isolation Forest algorithm.

- GeoIP Enrichment – enriches login attempts and sensitive actions with country, city, and ASN/ISP data using MaxMind GeoIP.
- Notifier – sends email notifications via SendGrid in cases of critical events such as configuration changes.

The application-layer modules rely on several types of storage located in the data layer:

- PostgreSQL – the primary relational database for persistent data (users, devices, login attempts).
- Redis – a cache and buffer for client-side events, as well as a Pub/Sub mechanism for key-expiration events.
- Markdown logs – human-readable summaries of user interactions generated by GenAI.
- Configuration files and keys – settings for external services (OpenRouter, SendGrid, MaxMind GeoIP) and elliptic-curve cryptographic keys provided through environment variables.

The architecture also includes two external services:

- OpenRouter – used to generate GenAI behavior summaries.
- SendGrid – used to send email notifications for anomalies and configuration changes.

3.3. Data Flow and Component Interaction

The information flow in Forest Gate can be summarized in the following steps:

1. *Session initialization.*

The user accesses the system through the client interface, after which the system generates a pre-auth cookie and a device fingerprint. These enable early device identification and the application of baseline rate limits.

2. *Authentication.*

The Auth Controller validates the provided credentials and creates a session, which is stored in Redis and PostgreSQL. Additionally, the accumulated contextual information (GeoIP data, fingerprint, timestamp, IP address) is passed to Isolation Forest, which determines whether the login attempt is normal or requires blocking or an additional authentication factor (MFA).

3. *Behavior monitoring.*

The Behavior Collector gathers contextual and temporal indicators related to user actions and stores them in Redis with a limited TTL.

4. *Anomaly detection.*

The Anomaly Detector analyzes the collected events using Isolation Forest and computes a real-time risk score.

5. *Summarization and interpretation.*

The Summary Engine listens for TTL expiration events from Redis (Pub/Sub), retrieves the accumulated logs, and sends them to GenAI via OpenRouter. The generative model produces a concise and readable summary of the behavior.

6. *Notification.*

When necessary, the Notifier Module triggers SendGrid and sends an alert to the administrator containing information about the behavior, the detected anomaly, and the recommended action.

7. *Visualization and follow-up actions.*

The data are stored and can be analyzed to detect long-term trends, recurring anomalies, or emerging attack patterns.

3.4. GenAI

An innovative characteristic of Forest Gate is the integration of GenAI as an active component within the security architecture. The goal is not to replace traditional mechanisms for analysis or decision-making, but to add a cognitive layer capable of interpreting and presenting complex system processes in a human-readable form.

In classical authentication models, logs and system events remain in a raw, technical format that is difficult to analyze by non-experts. As the volume and complexity of such data increase, there is a growing need for an intelligent module capable of extracting meaning, context, and trends from fragmented information. This is where GenAI becomes particularly relevant.

Within Forest Gate, GenAI operates as an explanatory and analytical component performing three main functions:

- Support during code development – used to generate, verify, and optimize parts of the Rust program logic through interactive communication with the developer.
- Log and event summarization – automatically transforms system records into concise, structured textual explanations suitable for auditing or for downstream training of security models.
- Support for automated alerting – formulates the content of administrator notifications in natural language, clearly distinguishing critical from non-critical events.

3.4.1. Code Generation, Log Summarization, and Automated Notifications

During the development of the system, GenAI was used to support various aspects of programming:

- Template code generation – modeling REST API structures, handling JSON requests, and implementing middleware layers in Rust.
- Refactoring and optimization – providing suggestions for optimizing functions, reducing

latency, and improving concurrency management through async/await.
- Security verification – automatically identifying potential vulnerabilities in the authentication logic, including incorrect token validation, uninitialized variables, or inconsistent access to shared resources.
- Documentation generation – automatically producing descriptions of functions, structures, and API routes in Markdown format, ensuring consistency between the codebase and the documentation.

One of the most important roles of GenAI in Forest Gate is the automatic summarization of system logs and session data. In traditional systems, administrators must manually review hundreds of log lines – a process that is often slow, labor-intensive, and prone to error. In Forest Gate, GenAI acts as a behavioral interpreter, analyzing the sequence of events within a session and transforming them into a concise and comprehensible textual summary. For example:

The user performed advanced search actions and filtered products by price and rating. They added multiple items to the cart and initiated checkout, then changed the shipping address. No anomalies detected. Risk level: Low.

A burst of rapid actions occurred, including repeated add-to-cart and remove actions and multiple coupon attempts. The user refreshed pages several times and toggled payment options. This is unusual volume but not inherently malicious. Risk level: Medium.

The user attempted multiple security bypasses. They tried CSRF attacks twice, followed by an injection attempt. They also attempted to upload a malicious file and routed their request through a proxy. Risk level: High.

These summaries are generated automatically through system and user instructions defined in the Rust Summary Engine module. The raw data from Redis and PostgreSQL are formatted into contextual JSON and sent to the OpenRouter API, whose response is stored in Markdown format. The resulting summaries are saved as log files and can be used both by administrators and for further analysis or model training.

The third functional role of GenAI relates to intelligent alerting and incident communication. When the Anomaly Detector identifies a deviation from normal behavior (e.g., multiple login attempts from different countries within a short interval), the system sends structured information to the Notifier module, which then triggers two processes:

1. Sending a concise technical alert via the SendGrid API containing the specific parameters of the deviation;
2. Generating a textual summary through GenAI that describes the situation in natural language, for example:

Atypical behavior detected for user g.atanasov: two access attempts from IP addresses in different countries within an 8-minute interval. Likelihood of account compromise: medium. Temporary token suspension is recommended.

3.4.2. System and User Prompts

The interaction between the system and GenAI is managed through two main types of instructions – system prompts and user prompts, which together define the model's role, objectives, and the format of the generated response.

The system prompt defines the model's primary role and its operational constraints. In Forest Gate, it is designed so that the model functions as an intelligent security analysis assistant, with restrictions aimed at producing concise and accurate summaries without hallucinations or fabricated information:

You are a security assistant integrated into the Forest Gate system.
Your role is to analyze user session logs and behavioral data.
Summarize the events objectively, in Markdown format, and avoid any assumptions.
Never include speculative statements or personal interpretations.
Be concise (3–5 sentences maximum) and focus on session context, anomalies, and risk level.

User prompts are generated dynamically during system operation. They contain an interaction identifier and a list of events, and are formatted in JSON. Examples of user prompts include:

```
{
  "interaction_id": "f8d2a13c-18ca-4782-88b6-a8bbe8f309de",
  "event": ["clicked add to cart button", "tried uploading bad file"]
}
{
  "interaction_id": "a1b2c3d4-e5f6-7890-abcd-1234567890aa",
  "event": ["visited /shop", "clicked add to cart on Soap", "tried CSRF"]
}
{
  "interaction_id": "e5f6a7b8-c9d0-1234-ef01-5678901234ee",
  "event": ["visited /login", "attempted login username=user1", "attempted login username=user1", "rate limited /login"]
}
```

4. Experiments

The conducted experiments aimed to evaluate the extent to which the Forest Gate prototype can operate stably, reliably, and in real-time under different user-behavior scenarios. The tests were performed in a controlled environment using Docker, Redis,

PostgreSQL, and NGINX, while Postman, Swagger, and LucidChart were used to simulate realistic workloads and visualize the processes.

4.1. Experimental Environment and Methodology

The system was deployed using Docker Compose, with each service (application, database, cache, and reverse proxy) running in its own isolated container.

Technologies and tools used:

- PostgreSQL – primary storage for users, sessions, and login attempts;
- Redis – event caching and implementation of the Pub/Sub mechanism for activity monitoring;
- NGINX – reverse proxy simulating production-like conditions;
- SendGrid API – for sending notifications upon configuration changes;
- MaxMind GeoIP – for geolocation enrichment and detection of "impossible travel" cases;
- OpenRouter (GenAI) – for summarizing sessions and generating textual reports;
- Postman and Swagger – for functional testing and API endpoint validation.

The simulations included several access scenarios:

- standard login and logout from a familiar device;
- multiple unsuccessful login attempts;
- access attempts from different geographical locations within a short time interval;
- high-frequency event generation by manipulating TTL keys in Redis.

4.2. Performance and Reliability

Experiments were conducted to evaluate the behavior of the Redis Pub/Sub mechanism, the efficiency of GeoIP analysis, and the system's performance under high load.

4.2.1. Redis Pub/Sub Operation

During stress tests involving numerous concurrent sessions, Redis demonstrated stable performance, with expired events being processed in under 50 ms. Excellent synchronization was observed between the background flusher process and the Markdown generator – upon TTL expiration, the system automatically extracted all recorded actions, removed them from the cache, and generated a concise summary using GenAI. No event retention issues or missed notifications were detected.

4.2.2. GeoIP Analysis Efficiency

The geolocation enrichment module (MaxMind) correctly identified locations with over 98% accuracy. "Impossible travel" scenarios (e.g., sequential logins from Sofia and Tokyo within 10 minutes) were successfully detected and reflected in the logs. The GenAI-generated summaries clearly highlighted such cases, for example:

> *Detected login from a distant location inconsistent with previous session. Possible credential sharing or VPN usage.*

4.2.3. High-Load Performance

The Rust backend maintained low latency under concurrent requests (35–40 ms per request with 100 simultaneous users). Latency was measured using the formula:

$$L_{avr} = \frac{1}{N}\sum_{i=1}^{N}(t_{out,i} - t_{in,i}), \quad (1)$$

where L_{avr} is the average latency (ms), N is the number of processed requests, and $t_{out,i}$ and $t_{in,i}$ are the completion and arrival times of the i-th request, respectively.

The use of async/await and Rust's memory safety mechanisms made it possible to avoid memory leaks and race conditions that are common in other languages. In simulations with 100 concurrent users, the calculated value was $L_{avr} \approx 37ms$, confirming the system's low latency and the efficiency of Rust's asynchronous request handling.

To measure throughput, the following formula was used:

$$T = \frac{N}{\Delta t}, \quad (2)$$

where T is the throughput (number of requests per second), N is the total number of processed requests, and Δt is the measurement time interval (in seconds).

In the performed experiments, when the /users/login endpoint processed 1,303 requests in 317.1 seconds, the measured throughput was $T = 4.11\,req/s$, which corresponds to service behavior with long-running requests.

To express the system's reliability under load, the classical definition of the success coefficient was used:

$$R = 1 - \frac{E_f}{E_t}, \quad (3)$$

where $R \in [0,1]$ is the system reliability, E_f is the number of failed or interrupted requests, and E_t is the total number of requests.

During testing of /users/login with 1,303 requests and 0 errors, the reliability was $R = 1.0000$, which demonstrates stable system behavior under load.

4.3. Summarizing Results with GenAI

The most interesting part of the experiments was related to the behavior of GenAI when processing logs. For each completed session, Redis forwarded

structured data to the Summary Engine module, which then sent a request to the OpenRouter API. In this context, GenAI can be viewed as a data-transformation function – from structured logs to a semantic summary. According to the classical formulation, this transformation can be expressed as:

$$S = \Phi(G, D) = arg \max_{s \in \Sigma} P(s|D, G), \quad (4)$$

where:

- D is the set of structured logs for a given session;
- G is the GenAI model parameterized through the system and user prompts;
- S is the resulting textual summary;
- Σ is the space of all possible textual outputs;
- $P(s \mid D, G)$ is the probability that a given sentence is the most faithful linguistic representation of the data.

The $arg\,max$ function returns the argument/sentence s for which the probability $P(s \mid D, G)$ is maximized. This formulation shows that the summary S is the result of optimizing the likelihood that the generated text expresses as fully and accurately as possible the structure of the data D given the context G.

During the experiments, the GenAI models used – ChatGPT and Claude – generated Markdown summaries with an average length of 4–6 sentences. The results showed that:

- 100% of the summaries were grammatically correct and easy to read;
- 91% correctly classified the behavior as normal or suspicious, consistent with the simulation scenario;
- 7% contained overly generalized conclusions (e.g., "activity appears normal" in cases of partial deviation);
- 2% were rejected by the system's validation due to missing the expected Markdown format.

These results confirm that the combination of stateless generation and automatic validation is an effective approach for achieving high accuracy without human intervention.

4.4. Behavioral Analysis Using Isolation Forest

For experimental demonstration, the Isolation Forest algorithm was used, trained on data generated by a GenAI script (with 15% anomalous records).

The results showed:

- Anomaly detection accuracy: approximately 94%;
- False positives (normal attempts classified as anomalies): below 6%;
- Strong dependence on the richness of behavioral data – longer sessions significantly increased accuracy.

The conclusion is that integrating the algorithm in real-time would allow the system to compute a probabilistic risk score for each session and trigger GenAI summaries with varying levels of detail.

5. Discussion

The developed model offers the following advantages:

- It transforms technical logs into concise and comprehensible GenAI summaries, supporting administrators and accelerating incident response.
- The reactive architecture (Redis Pub/Sub, TTL) ensures low latency and real-time processing.
- Integrating behavioral and geographical factors adds an additional layer of security.
- Experiments with the prototype confirm good performance and stability.

The main limitations include:

- Dependence on external GenAI services, which introduces risks related to privacy, latency, and resilience; transitioning to local models is a potential solution.
- The possibility that GenAI may generate variable or incorrect interpretations.
- Ethical and legal risks, including the need for GDPR compliance.
- The requirement for human verification in critical decisions.

Effective use of GenAI in security requires a clear separation of roles:

- GenAI analyzes and explains;
- The system reacts automatically within predefined boundaries;
- The human operator makes the final decision.

This three-level organization prevents excessive autonomy of artificial intelligence and ensures ethical and legal reliability in security processes.

In addition to improving the prototype, another direction in the development of Forest Gate is its integration with artificial neural networks. Based on 150,000 accumulated login events, experiments were conducted to predict likely geolocations for future access attempts. Using the Orange Data Mining environment, LSTM Autoencoder models were trained on normal login sequences for individual geo-clusters. The network learns to reconstruct typical login patterns, and deviations are interpreted as indicators of elevated risk.

After training on 150,000 samples, the model predicted the geolocations of the next 2,000 login attempts. The results show:

- Bulgaria: 226 expected attempts with approximately 77% probability of successful access (Figure 2).

- Seychelles: 35 attempts with approximately 6% probability of success.
- China/Hong Kong: 38 attempts with 0% predicted likelihood of successful access.

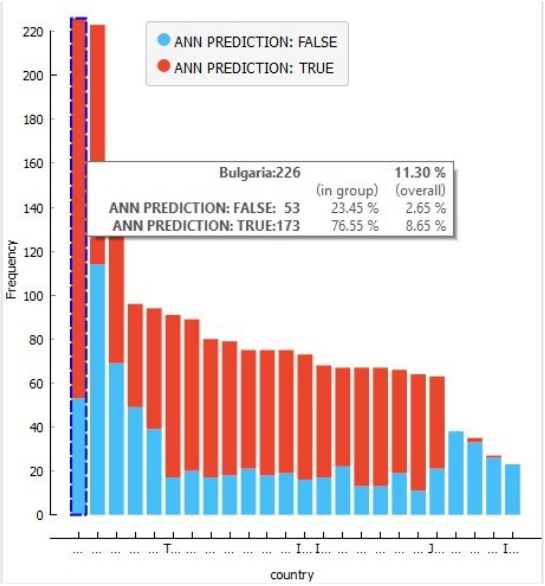


Figure 2. Probability of geolocations during login attempts according to the neural network.

The results can be visualized on a geographical map (Figure 3), which shows the global forecast of success or failure for the next 2,000 login attempts.

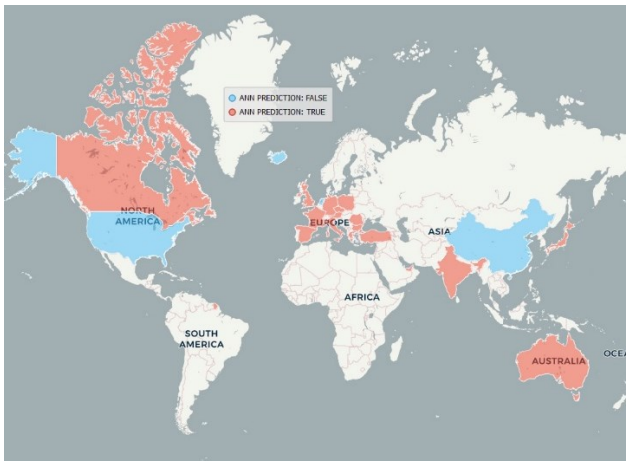


Figure 3. Geographical map of the predicted geolocations for the next 2,000 login attempts, according to the neural network.

6. Conclusion

This paper presented an experimental prototype of a secure authentication and user-session monitoring system built on the synergy of behavioral analysis, GenAI, and event-driven architecture. The Forest Gate project demonstrates that even at an early stage, it is possible to achieve effective, explainable, and adaptive access protection for web applications using lightweight and open technologies.

The experimental results show that GenAI can be integrated without degrading the performance or stability of authentication systems. The prototype demonstrates several advantages, including enhanced security, the transformation of technical data into comprehensible summaries, and accelerated detection and notification of critical events. At the same time, it must be acknowledged that the use of GenAI introduces risks related to the generation of inaccurate or hallucinated conclusions.

In this context, Forest Gate can be viewed as a next-generation hybrid platform that combines the power of language models with the predictability of engineered mechanisms. The approach is applicable not only in academic and experimental environments but also as a foundation for extensions into corporate or cloud-based solutions.

References

[1] Manyika, J., & Roxburgh, C. (2011). *The great transformer: The impact of the Internet on economic growth and prosperity*. McKinsey Global Institute, 1(0360-8581), 1-9.

[2] OECD. (2023). *Country digital education ecosystems and governance: A companion to Digital Education Outlook*. OECD Publishing. https://doi.org/10.1787/906134d4-en

[3] Liu, Y., & Qiang, Z. (2023). *Digital progress and trends report 2023* (English). World Bank Group. Retrieved from http://documents.worldbank.org/curated/en/099031924192524293

[4] Sahin, S., & Li, F. (2021). Don't forget the stuffing! Revisiting the security impact of typo-tolerant password authentication. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. https://doi.org/10.1145/3460120.3484791

[5] Otta, S. P., et al. (2023). A systematic survey of multi-factor authentication for cloud infrastructure. *Future Internet, 15*(4), 146. https://doi.org/10.3390/fi15040146

[6] Rabzelj, M., & Sedlar, U. (2025). Beyond the leak: Analyzing the real-world exploitation of stolen credentials using honeypots. *Sensors, 25*(12), 3676. https://doi.org/10.3390/s25123676

[7] Matzen, A., et al. (2025). Challenges and potential improvements for passkey adoption—A literature review with a user-centric perspective. *Applied Sciences, 15*(8), 4414. https://doi.org/10.3390/app15084414

[8] Verizon (DBIR). (2024). *Data breach investigations report*. Retrieved November 8, 2025, from https://www.verizon.com/business/resources/reports/2024-dbir-data-breach-investigations-report.pdf

[9] Baig, A. F., & Eskeland, S. (2021). Security, privacy, and usability in continuous authentication: A survey. *Sensors, 21*(17), 5967. https://doi.org/10.3390/s21175967

[10] Verma, A., Moghaddam, V., & Anwar, A. (2022). Data-driven behavioural biometrics for continuous and adaptive user verification using smartphone and smartwatch. *Sustainability, 14*(12), 7362. https://doi.org/10.3390/su14127362

[11] Papaioannou, M., et al. (2023). A survey on quantitative risk estimation approaches for secure and usable user authentication on smartphones. *Sensors, 23*(6), 2979. https://doi.org/10.3390/s23062979

[12] Bonneau, J., et al. (2015). Passwords and the evolution of imperfect authentication. *Communications of the ACM, 58*(7), 78–87. https://doi.org/10.1145/2699390

[13] Ometov, A., et al. (2018). Multi-factor authentication: A survey. *Cryptography, 2*(1), 1. https://doi.org/10.3390/cryptography2010001

[14] Hasan, S. S., et al. (2025). A review on secure authentication mechanisms for mobile security. *Sensors, 25*(3), 700. https://doi.org/10.3390/s25030700

[15] IBM. (2024). *Cost of a data breach report 2025: The AI oversight gap – Top insights, AI impact, and risk reduction best practices*. IBM. Retrieved November 8, 2025, from https://ibm.webcasts.com/starthere.jsp?ei=1726278

[16] Bułat, R., & Ogiela, M. R. (2023). Personalized context-aware authentication protocols in IoT. *Applied Sciences, 13*(7), 4216. https://doi.org/10.3390/app13074216

[17] Ryu, R., Yeom, S., Herbert, D., & Dermoudy, J. (2023). A comprehensive survey of context-aware continuous implicit authentication in online learning environments. *IEEE Access, 11*, 24561–24573. https://doi.org/10.1109/ACCESS.2023.3253484

[18] Saleem, J., et al. (2025). Machine learning-enhanced attribute-based authentication for secure IoT access control. *Sensors, 25*(9), 2779. https://doi.org/10.3390/s25092779

[19] de Almeida, M. G., & Canedo, E. D. (2022). Authentication and authorization in microservices architecture: A systematic literature review. *Applied Sciences, 12*(6), 3023. https://doi.org/10.3390/app12063023

[20] Sharif, A., et al. (2022). The eIDAS regulation: A survey of technological trends for European electronic identity schemes. *Applied Sciences, 12*(24), 12679. https://doi.org/10.3390/app122412679

[21] Han, A. H., & Lee, D. H. (2023). Detecting risky authentication using the OpenID Connect token exchange time. *Sensors, 23*(19), 8256. https://doi.org/10.3390/s23198256

[22] Venčkauskas, A., et al. (2023). Enhancing microservices security with token-based access control method. *Sensors, 23*(6), 3363. https://doi.org/10.3390/s23063363

[23] Wang, Y., Zhang, X., & Hu, H. (2023). Continuous user authentication on multiple smart devices. *Information, 14*(5), 274. https://doi.org/10.3390/info14050274

[24] Bansal, P., & Ouda, A. (2024). Continuous authentication in the digital age: An analysis of reinforcement learning and behavioral biometrics. *Computers, 13*(4), 103. https://doi.org/10.3390/computers13040103

[25] Wang, S., et al. (2021). Machine learning in network anomaly detection: A survey. *IEEE Access, 9*, 152379–152396. https://doi.org/10.1109/ACCESS.2021.3126834

[26] Kyaw, P. H., Gutierrez, J., & Ghobakhlou, A. (2024). A systematic review of deep learning techniques for phishing email detection. *Electronics, 13*(19), 3823. https://doi.org/10.3390/electronics13193823

[27] Wilk-Jakubowski, J. L., et al. (2025). Machine learning and neural networks for phishing detection: A systematic review (2017–2024). *Electronics, 14*(8), 3744. https://doi.org/10.3390/electronics14183744

[28] Balogh, Š. (2024). Using generative AI models to support cybersecurity analysts. *Electronics, 13*(23), 4718. https://doi.org/10.3390/electronics13234718

[29] Lee, J., et al. (2025). LogRESP-Agent: A recursive AI framework for context-aware log anomaly detection and TTP analysis. *Applied Sciences, 15*(13), 7237. https://doi.org/10.3390/app15137237

[30] Palma, G., et al. (2025). Leveraging large language models for scalable and explainable cybersecurity log analysis. *Journal of Cybersecurity and Privacy, 5*(3), 55. https://doi.org/10.3390/jcp5030055

[31] Zhu, S., et al. (2022). An optimized algorithm for dangerous driving behavior identification based on unbalanced data. *Electronics, 11*(10), 1557. https://doi.org/10.3390/electronics11101557

[32] Chen, J., et al. (2023). An anomaly detection method for wireless sensor networks based on the improved isolation forest. *Applied Sciences, 13*(2), 702. https://doi.org/10.3390/app13020702

[33] Villegas-Ch., et al. (2023). IoT anomaly detection to strengthen cybersecurity in the critical infrastructure of smart cities. *Applied Sciences, 13*(19), 10977. https://doi.org/10.3390/app131910977

[34] Chua, W., et al. (2024). Web traffic anomaly detection using isolation forest. *Informatics, 11*(4), 83. https://doi.org/10.3390/informatics11040083

[35] Alboqmi, R., & Gamble, R. F. (2025). Enhancing microservice security through vulnerability-driven trust in the service mesh architecture. *Sensors, 25*(3), 914. https://doi.org/10.3390/s25030914

[36] Mushtaq, S., Mohsin, M., & Mushtaq, M. M. (2025). A systematic literature review on the implementation and challenges of zero trust architecture across domains. *Sensors, 25*(19), 6118. https://doi.org/10.3390/s25196118