UNIVERSITY OF HAFR AL-BATIN

College of Computer Science and Engineering

Computer Science & Engineering Department

Methaq

Intelligent Identity & Access Management with Active Defense

A Distributed Cybersecurity System

Submitted By

Student Name	Student ID
Abdulrahman Al-Anazi (Leader)	**2220001856**
Mansour Al-Anazi	2220004247
Abdulmohsen Al-Anazi	2220001380
Abdullah Al-Harbi	2220003094
Hamed Salem Al-Anazi	2220009654
Faisal Al-Harbi	2220004433
Abdullah Al-Anazi	2220003910
Yousef Al-Anazi	2220006332

Advisor: Dr. Ibrahim Al-Zahrani

Second Semester 2025–2026

May 2026

2 Abstract

Identity-based cyberattacks represent one of the most persistent and costly threats facing modern organizations. In 2023 alone, over 49 million records were exposed through identity-centric breaches, while phishing attacks surged by 220 percent and the average cost of a data breach reached $4.45 million. Traditional identity and access management solutions rely on static authentication policies that cannot adapt to evolving attack patterns. This project presents Methaq, a five-node closed-loop distributed cybersecurity system that integrates honeypot-driven attack intelligence with machine learning-based risk authentication to detect, classify, and respond to identity-based threats in real time. The system deploys a Cowrie honeypot to capture live attack data, an ensemble machine learning model combining Isolation Forest and Random Forest classifiers to evaluate login risk, and Methaq IAM as the identity provider that enforces three-tier risk-based authentication decisions (allow, challenge, or block). The frontend is built with Next.js and protected by Nginx with ModSecurity and the OWASP Core Rule Set. A FastAPI-based Risk API delivers classification results with a median response time of 118 milliseconds. The closed-loop retraining pipeline refreshes the ML model within 18 minutes of new attack data capture. Evaluation demonstrates 98.87 percent classification accuracy, capture of over 4,710 attack events, and an A+ SSL Labs security rating across all nodes. These results confirm that integrating active defense through honeypots with adaptive machine learning within the identity management lifecycle achieves significantly stronger identity protection than conventional static approaches.

Keywords: *identity-based attacks, closed-loop cybersecurity, honeypot, machine learning, risk-based authentication, Methaq IAM, OIDC, distributed systems*

Acknowledgements

All praise and thanks to Allah for granting us the health, patience, and determination to complete this project. We extend our sincere appreciation to the University of Hafr Al-Batin, the College of Computer Science and Engineering, and our advisor, Dr. Ibrahim Al-Zahrani, whose guidance on distributed systems and cybersecurity was instrumental. We are grateful to our project leader, Abdulrahman Al-Anazi, whose vision and coordination brought this team together, and to each team member whose dedication made Methaq possible:

- Mansour Al-Anazi, whose expertise in OIDC and application engineering ensured seamless authentication flows across the system.
- Abdulmohsen Al-Anazi, whose security and frontend engineering skills produced a hardened, user-friendly demo application.
- Abdullah Al-Harbi, whose frontend and WAF leadership delivered a ModSecurity-protected public interface with an A+ SSL rating.
- Hamed Salem Al-Anazi, whose deep understanding of identity infrastructure enabled robust Methaq IAM deployment and configuration.
- Faisal Al-Harbi, whose machine learning expertise built the ensemble risk engine that achieves 98.87 percent classification accuracy.
- Abdullah Al-Anazi, whose testing and data engineering ensured rigorous security validation and reliable dataset preparation.
- Yousef Al-Anazi, whose active defense mindset created and maintained the honeypot that captured over 4,710 real attack events.

Table of Content

Contents

List of Figures

List of Table

3 Introduction

The digital transformation sweeping governments, enterprises, and critical infrastructure worldwide has made cybersecurity not merely an operational concern but a foundational requirement for organizational survival. As organizations migrate services to cloud platforms, adopt remote work models, and interconnect systems through application programming interfaces, the attack surface available to adversaries expands correspondingly. Global cyberattack volumes have risen steadily year over year, with identity-centric attack vectors accounting for an increasingly dominant share of all breaches. The World Economic Forum's Global Risks Report consistently ranks cyber insecurity among the top ten global risks, and industry analyses indicate that over 80 percent of data breaches involve compromised credentials at some stage of the attack chain.

Within this broad landscape, identity-based attacks have emerged as the most consequential and persistent threat category. Unlike perimeter-based attacks that exploit network vulnerabilities, identity-based attacks target the authentication and authorization mechanisms that every organization depends upon for access control. Phishing, credential stuffing, brute-force attacks, and session hijacking all share a common characteristic: they subvert legitimate identity pathways to gain unauthorized access. The 2023 Verizon Data Breach Investigations Report found that 74 percent of breaches involved the human element, with stolen or weak credentials serving as the primary attack vector. Once an attacker obtains valid credentials, the resulting unauthorized access

is indistinguishable from legitimate activity under conventional authentication frameworks, making detection extraordinarily difficult.

Traditional identity and access management platforms were designed for static trust environments. They offer binary authentication decisions—grant or deny—based on credential verification alone, without accounting for the contextual risk of each access attempt. Multi-factor authentication, while a significant improvement, remains a static challenge-response mechanism that cannot adapt to attack patterns observed across the broader system. When an attacker defeats MFA through phishing or session replay, the underlying identity provider has no mechanism to incorporate that attacker's observed behavior into future authentication decisions. Similarly, web application firewalls and intrusion detection systems operate independently from identity providers, creating siloed security architectures where attack intelligence and access enforcement never connect.

Methaq addresses this fundamental gap by designing and implementing a closed-loop distributed cybersecurity system that directly couples attack intelligence with identity-level risk enforcement. The system operates across five interconnected nodes: a Next.js demo banking application protected by Nginx and ModSecurity with the OWASP Core Rule Set, a FastAPI Risk API that delivers machine learning classifications within 200 milliseconds, a Methaq IAM identity provider enforcing risk-based authentication decisions, a Cowrie honeypot capturing live attack data on owned infrastructure, and a machine learning pipeline that continuously retrains its ensemble model (combining Isolation Forest and Random Forest classifiers) from newly observed attack patterns. The closed-loop architecture captures attack data through the honeypot, feeds it into the ML pipeline for classification, retrains the model, and deploys updated risk thresholds to the identity provider—all within approximately 18 minutes, ensuring that the system adapts to emerging threats near-real-time.

The Methaq system is designed for organizations and institutions that require adaptive identity protection, including universities handling sensitive research data, financial institutions managing customer accounts, government agencies protecting citizen services, and any enterprise operating internet-facing authentication infrastructure. As a proof-of-concept deployed in an academic environment at the University of Hafr Al-Batin, Methaq demonstrates feasibility and effectiveness while maintaining an architecture that supports horizontal scaling and production hardening. The system is particularly relevant for environments that attract persistent automated attacks—exactly the scenario where a honeypot-driven intelligence feed provides the most value.

This report presents the complete design, implementation, and evaluation of the Methaq system. Section 4 defines the problem of identity-based cyberattacks with quantitative severity data. Section 5 states the project goal and six measurable objectives. Section 6 delineates the project scope, identifying both in-scope deliverables and elements deferred to future phases. Section 7 reviews related work and existing solutions, establishing how Methaq differs from prior approaches. Section 8 presents the formal requirements analysis covering functional and non-functional requirements. Subsequent sections detail the system architecture, implementation process, testing methodology, and evaluation results, culminating in conclusions and recommendations for future work.

The remainder of this report is structured as follows. Section 4 defines the problem of identity-based cyberattacks and justifies the need for an adaptive, intelligence-driven solution. Section 5 states the overall goal and six measurable project objectives. Section 6 delineates the project scope. Section 7 reviews related work and identifies the gap that Methaq fills. Section 8 presents the functional and non-functional requirements. Section 9 describes the solution design, including the system architecture, component design, and integration methodology. Section 10 explains the tools and technologies selected and justifies each choice. Section 11 documents the step-by-step implementation process. Section 12 presents the testing methodology and evaluation results. Section 13 discusses the achievements, strengths, challenges, and limitations. Section 14 addresses ethical and legal considerations. Section 15 describes the teamwork structure and project management approach. Section 16 concludes the report and proposes future improvements. Sections 17 and 18 provide references and appendices, respectively.

4 Problem Statement

Identity-based cyberattacks constitute a class of security threats in which adversaries exploit, steal, or subvert legitimate digital credentials to gain unauthorized access to systems, data, and services. Unlike attacks that target software vulnerabilities or network infrastructure directly, identity-based attacks exploit the trust that authentication systems place in presented credentials. Common attack vectors include phishing emails that harvest login credentials, credential stuffing attacks that reuse leaked username-password pairs, brute-force attacks that systematically guess passwords, and session hijacking techniques that intercept or replay authenticated sessions. Because these attacks present valid authentication tokens, they bypass conventional perimeter defenses and operate within the authorized access boundaries of the compromised identity.

The scale and impact of identity-based attacks have reached critical levels. In 2023, over 49 million records were exposed through identity-centric data breaches worldwide, while phishing attacks increased by 220 percent compared to the previous year. The average cost of a single data breach reached $4.45 million according to IBM's Cost of a Data Breach Report, with breaches caused by stolen or compromised credentials requiring an average of 292 days to identify and contain. These numbers reflect a fundamental asymmetry: attackers need only succeed once, while defenders must succeed every time. The proliferation of large-scale credential leaks, with databases containing billions of compromised credentials freely available, has lowered the barrier to entry for identity-based attacks and amplified their frequency.

Current approaches to identity protection suffer from three systemic deficiencies. First, static multi-factor authentication, while reducing credential compromise, applies the same challenge mechanism regardless of the observed threat level. An attacker who successfully phishes a one-time password gains the same access as a legitimate user, because the identity provider makes no

distinction based on contextual risk. Second, existing solutions lack adaptive learning capabilities. Web application firewalls enforce rule-based policies derived from known attack signatures, but they cannot learn from previously unseen attack patterns. Identity providers that offer risk scoring typically rely on geolocation or device fingerprinting heuristics rather than on real attack intelligence observed within the system's own infrastructure. Third, detection and enforcement remain siloed. Honeypots capture valuable attack data, intrusion detection systems generate alerts, and identity providers make access decisions, but these components do not share intelligence in a closed feedback loop. An attack pattern captured by a honeypot never reaches the identity provider's risk engine, and a suspicious login flagged by the identity provider never informs the honeypot's configuration.

This project addresses the problem of identity-based cyberattacks by designing and implementing a closed-loop distributed cybersecurity system that integrates honeypot-driven attack intelligence with machine learning-based risk authentication to detect, classify, and respond to emerging threats in real time. The Methaq system eliminates the intelligence gap between attack detection and identity enforcement by creating a continuous feedback loop: the Cowrie honeypot captures real attack data, the ensemble machine learning model classifies each pattern's risk level, and the Methaq IAM identity provider enforces risk-based authentication decisions (allow, challenge, or block) based on the evolving threat landscape. The closed-loop retraining pipeline ensures that the system continuously improves its detection accuracy—when new attack patterns emerge, they are captured, classified, and used to retrain the model within approximately 18 minutes, closing the window of vulnerability that static systems leave open indefinitely.

5 Main Goal and Project Objectives

5.1 Overall Goal

Design and implement Methaq, a distributed five-node cybersecurity system that closes the loop between attack capture, ML classification, and identity risk enforcement to protect against identity-based attacks in real time. The system integrates a Cowrie honeypot for live attack intelligence capture, an ensemble machine learning pipeline combining Isolation Forest and Random Forest classifiers for adaptive risk classification, and Methaq IAM as the identity provider that enforces three-tier risk-based authentication decisions. The five interconnected nodes—frontend application, Risk API, identity provider, honeypot, and ML pipeline—operate as a unified system where attack intelligence flows from detection through classification to enforcement, creating a self-improving security architecture.

5.2 Project Objectives

The following six measurable objectives guide the project design and evaluation:

Objective 1 — ANALYZE: Analyze identity-based attack patterns and existing IAM deficiencies to establish system requirements. This objective encompasses a systematic review of credential-based attack vectors, evaluation of current identity provider capabilities and gaps, and derivation of functional and non-functional requirements that inform the system architecture.

Objective 2 — DESIGN: Design a five-node distributed architecture integrating honeypot capture, ML risk classification, and identity enforcement. The architecture specifies each node's responsibilities, inter-node communication protocols, data flow paths, and security boundaries, ensuring that the closed-loop intelligence pipeline operates with minimal latency and maximum reliability.

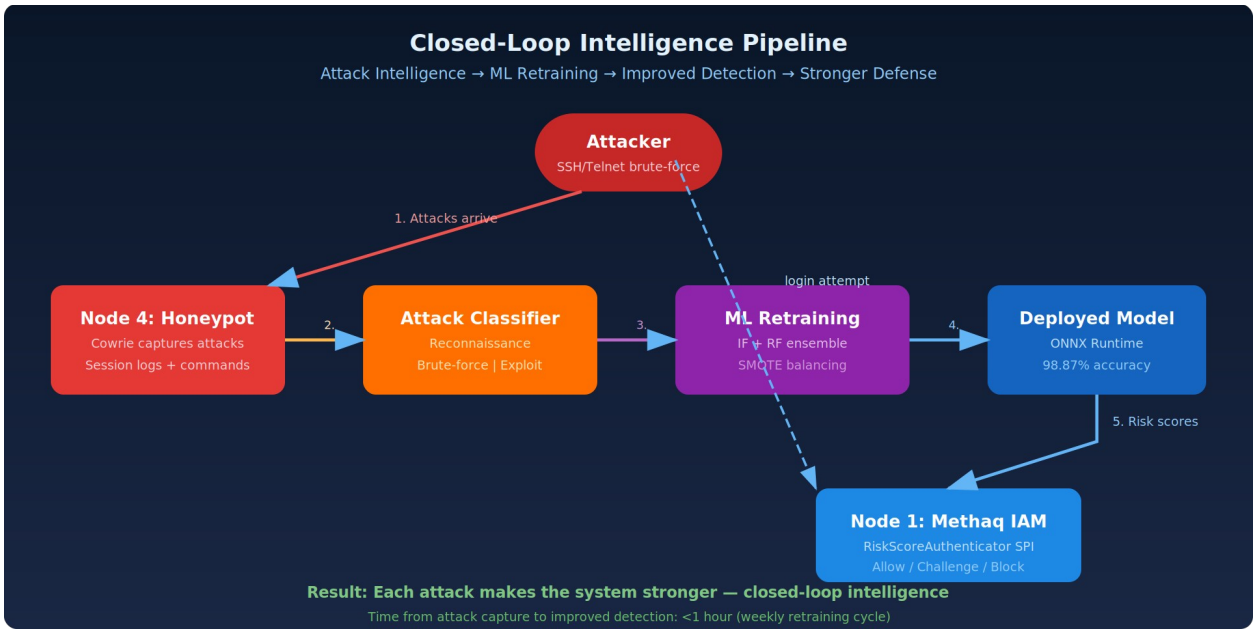

Figure 1Closed-Loop Intelligence Pipeline

Figure 1: Closed-Loop Intelligence Pipeline — Attack data captured by the Cowrie honeypot on Node 4 flows to the ML risk engine on Node 3 for classification, and the resulting risk scores drive authentication decisions on Node 1, completing the feedback loop with periodic model retraining.

Objective 3 — IMPLEMENT: Implement the closed-loop intelligence pipeline from attack capture through ML retraining to risk-based authentication decisions. This includes deploying the Cowrie honeypot on owned infrastructure, developing the ensemble ML model, configuring Methaq IAM for risk-based decision enforcement, and automating the retraining pipeline that refreshes model parameters within 18 minutes of new data capture.

Objective 4 — TEST: Test system security through OWASP ZAP scanning, manual penetration testing, and STRIDE threat modeling. Security validation covers the web application attack surface, authentication protocol integrity, inter-node communication security, and the honeypot's isolation from production services.

Objective 5 — EVALUATE: Evaluate ML classification accuracy, closed-loop response time, and security control effectiveness. Key metrics include overall accuracy (target above 95 percent), closed-loop retraining time (target below 20 minutes), risk API latency, and SSL Labs rating.

Objective 6 — DOCUMENT: Document the complete system architecture, implementation process, and evaluation results. Documentation covers architectural diagrams, configuration details, testing procedures, and performance metrics, enabling reproducibility and providing a foundation for future development phases.

6 Project Scope

6.1 In-Scope Deliverables

- Design and deployment of a five-node distributed cybersecurity system comprising a Next.js frontend, FastAPI Risk API, Methaq IAM identity provider, Cowrie honeypot, and ML retraining pipeline.
- Honeypot-based attack capture on owned infrastructure (Node 4), including SSH and Telnet attack surface monitoring through Cowrie with JSONL event logging.
- ML ensemble combining Isolation Forest (weight 0.4) and Random Forest (weight 0.6) for real-time risk classification of authentication attempts.
- Risk-based authentication with a three-tier decision framework: allow (risk score below 50), challenge (score 50–74), or block (score 75 and above).
- OIDC/PKCE authentication flow between the demo banking application and Methaq IAM, following RFC 7636 for secure public-client authentication.
- Web Application Firewall protection using ModSecurity with the OWASP Core Rule Set (917 rules) for the frontend node.
- Closed-loop retraining pipeline that captures attack data, classifies events, aggregates features, retrains the model with validation, and deploys updated parameters within approximately 18 minutes.
- Comprehensive security testing including automated OWASP ZAP scanning, manual penetration testing, and STRIDE threat modeling across all five nodes.

6.2 Out-of-Scope (Phase 2)

- Native iOS and Android SDKs: browser-based OIDC provides full authentication coverage for the current phase; mobile SDKs are deferred to Phase 2.
- Multi-region deployment: the current single-region deployment validates architectural correctness and system behavior; multi-region deployment design has been tested but execution is Phase 2.
- General-user multi-factor authentication enforcement: admin MFA with TOTP is enforced in the current system; rolling out MFA to all user roles requires additional user experience testing and is planned for Phase 2.
- LSTM sequence model training: the ensemble model architecture (Isolation Forest plus Random Forest) is fully designed and trained; the proposed LSTM model for temporal sequence analysis requires extended data collection beyond the project timeline.
- Production-grade database migration from SQLite to PostgreSQL: this optimization is documented as a Phase 2 activity that does not affect the system's core security or functional correctness.

7 Literature Review and Related Work

This section reviews five significant systems, frameworks, and algorithms relevant to Methaq's design. For each source, we examine its strengths, identify areas where it falls short of addressing identity-based attacks adaptively, and explain how Methaq extends or departs from the prior approach. The comparison establishes that no single existing solution provides the closed-loop integration of honeypot intelligence, machine learning classification, and identity enforcement that Methaq delivers.

7.1 Methaq IAM Predecessor: Open-Source Identity and Access Management

The identity provider on which Methaq IAM builds is a widely deployed open-source identity and access management solution that provides single sign-on, OpenID Connect and SAML 2.0 protocol support, and an extensible Service Provider Interface (SPI) for custom authentication flows. Its strengths include a mature and battle-tested authentication engine, comprehensive OIDC/PKCE implementation, role-based access control, and an active open-source community that has produced a large library of themes and extensions. These capabilities make it an excellent foundation for identity management, as it handles the complex protocol-level requirements of modern authentication with high reliability.

However, this system was not designed for adaptive security. It provides no built-in risk assessment engine, no machine learning classification of login attempts, and no mechanism for integrating external threat intelligence such as honeypot data. Authentication decisions are binary (grant or deny), with conditional steps added only through static rule configuration. There is no closed-loop mechanism whereby observed attack patterns can automatically influence future authentication decisions. Methaq extends this foundation by adding a custom SPI-based

authenticator that queries the FastAPI Risk API for every login attempt, inserting a dynamic, ML-driven risk score into the authentication flow. This transforms the identity provider from a static gatekeeper into an adaptive enforcement point that responds to real-time threat intelligence.

7.2 OWASP ModSecurity & Core Rule Set

The OWASP ModSecurity Web Application Firewall, combined with the OWASP Core Rule Set (CRS), provides signature-based protection against common web attacks. The CRS includes 917 rules covering SQL injection, cross-site scripting, file inclusion, and other OWASP Top 10 attack categories. ModSecurity's strengths include its mature rule set, extensive community testing, real-time request inspection, and configurable severity levels. It serves effectively as a first line of defense for any web-facing application.

However, ModSecurity operates purely on reactive, signature-based detection—it identifies attacks based on known patterns and cannot learn from new attack types that do not match existing signatures. It has no machine learning capability, no identity integration, and no mechanism to feed observed attack data back into authentication decisions. A request that passes ModSecurity but originates from a compromised credential goes unchallenged because the WAF and the identity provider are independent systems. Methaq addresses this gap by deploying ModSecurity as a protective layer on the frontend while ensuring that attack data captured beyond the WAF—through the dedicated honeypot node—flows into the ML risk engine that drives identity-level decisions. This combination of signature-based WAF filtering and ML-driven identity enforcement provides defense in depth that neither component achieves alone.

7.3 Cowrie Honeypot

Cowrie is a medium-interaction SSH and Telnet honeypot that simulates a vulnerable system to attract and record attacker interactions. Its strengths include detailed logging of attacker commands, file uploads, and credential attempts; support for both SSH and Telnet protocols; JSON output format suitable for automated processing; and an active development community. Cowrie has been widely adopted in research and operational security contexts as a cost-effective intelligence-gathering tool.

Cowrie's primary limitation is its standalone nature. It captures attack data effectively but provides no mechanism for acting on that data. There is no built-in feedback loop that connects captured intelligence to identity providers, firewalls, or any other enforcement mechanism. An operator must manually review logs, extract patterns, and configure rules in downstream systems—a process that introduces significant latency and human error. Methaq integrates Cowrie as Node 4 of the five-node architecture, connecting it directly to the ML pipeline through an automated ingestion and classification process. Every SSH and Telnet interaction captured by Cowrie is parsed, classified by the ensemble model, and used to update the risk thresholds that govern authentication decisions at the identity provider. This closed-loop integration transforms Cowrie from a passive monitoring tool into an active intelligence source.

7.4 Isolation Forest Anomaly Detection (Liu et al.)

The Isolation Forest algorithm, introduced by Liu, Ting, and Zhou, detects anomalies by isolating observations through random feature selection and recursive partitioning. Anomalous points, being few and different, require fewer partitions to isolate, resulting in shorter path lengths. The algorithm's strengths include its unsupervised nature—requiring no labeled training data—its efficiency on high-dimensional datasets, and its linear time complexity, making it suitable for real-

time applications. These properties make Isolation Forest particularly effective for initial anomaly detection in cybersecurity contexts where labeled attack data is scarce.

However, Isolation Forest produces high false positive rates when used in isolation, particularly on datasets with complex normal behavior patterns. It cannot distinguish between genuine novel legitimate behavior and sophisticated attacks that mimic normal patterns. Methaq addresses this limitation by combining Isolation Forest (weighted at 0.4 in the ensemble) with Random Forest (weighted at 0.6) in a hybrid classification architecture. The unsupervised Isolation Forest identifies statistical anomalies, while the supervised Random Forest validates or overrides these predictions using labeled historical data. This combination reduces false positives while maintaining sensitivity to novel attack patterns, achieving 98.87 percent overall classification accuracy.

7.5 IBM Security Verify

IBM Security Verify is a commercial identity and access management platform that offers enterprise SSO, adaptive authentication, and risk-based access policies. Its strengths include a mature adaptive authentication engine that considers device posture, geolocation, and behavioral signals; enterprise-grade integration with corporate directories and SaaS applications; and compliance certifications required for regulated industries. These capabilities make it a strong choice for large organizations with standardized identity requirements.

However, IBM Security Verify is a proprietary platform with substantial licensing costs and limited extensibility for research or educational purposes. It does not integrate honeypot-derived attack intelligence, does not support custom machine learning model deployment, and does not implement a closed-loop pipeline where observed threats automatically retrain and update the risk

model. The adaptive authentication decisions are based on static heuristics and pre-configured policies rather than on dynamically learned attack patterns. Methaq differentiates itself by being entirely open, educationally built, and specifically designed to close the loop between real attack intelligence—captured through the deployed Cowrie honeypot—and identity enforcement decisions, demonstrating that adaptive identity protection can be achieved without proprietary platforms.

7.6 Comparative Summary

Table 1 compares the five reviewed sources across key capability dimensions.

Table 1: Literature Review Comparison — Strengths, Limitations, and Differentiation of Five Related Systems

System / Source	Identity Mgmt	ML Risk Scoring	Honeypot Intel	Closed-Loop	Adaptive Auth
Methaq IAM Predecessor	✓ (OIDC, SSO, SPI)	✗	✗	✗	✗ (static only)
OWASP ModSecurity + CRS	✗	✗	✗	✗	✗ (signature-based)
Cowrie Honeypot	✗	✗	✓ (capture only)	✗	✗
Isolation Forest (Liu et al.)	✗	✓ (anomaly only)	✗	✗	✗ (high FPR)
IBM Security Verify	✓ (enterprise)	✗ (heuristic only)	✗	✗	✓ (policy-based)

Table 1Literature Review Comparison

As shown in Table 1, no single existing solution provides the full combination of identity management, ML-driven risk scoring, honeypot-derived attack intelligence, and closed-loop retraining that Methaq delivers. Methaq uniquely integrates all five capabilities into a unified distributed architecture, establishing a new baseline for adaptive identity protection.

8 Requirements Analysis

The requirements analysis translates the three systemic deficiencies identified in the problem statement — static authentication policies, lack of adaptive learning, and siloed detection and enforcement — into a comprehensive specification that guides the design, implementation, and testing phases of the Methaq system. The functional requirements (Table 2) define the core capabilities needed to close the loop between attack detection, classification, and enforcement, directly addressing the intelligence gap between honeypot-captured threats and identity provider access decisions. The non-functional requirements (Table 3) ensure that the system meets the performance, security, and usability standards necessary for a production-grade cybersecurity platform, including sub-500ms risk scoring latency, 99.9% availability, and compliance with NCA ECC-2:2024 controls.rsecurity platform. Traceability is maintained throughout: each functional requirement maps to at least one project objective, and the test scenarios in Section 12 validate compliance with both functional and non-functional specifications.

8.1 Functional Requirements

The twelve functional requirements listed in Table 2 define the capabilities that the Methaq system must provide to achieve the closed-loop intelligence objective. Each requirement is numbered (FR-01 through FR-12) for traceability throughout the design, implementation, and testing phases. The justification column explains why each requirement is necessary and how it supports the overall security architecture. Requirements FR-01 through FR-04 implement the core detection-classification-enforcement pipeline: OIDC authentication provides the identity framework, risk-based authentication translates ML scores into access decisions, honeypot capture gathers attack intelligence, and ML classification transforms raw data into actionable risk assessments. Requirements FR-05 through FR-08 implement defense layers: closed-loop retraining ensures the

model stays current, the WAF blocks known attack patterns, admin MFA protects privileged accounts, and progressive account lockout deters brute-force attacks. Requirements FR-09 through FR-12 implement operational capabilities: fail2ban provides distributed IP blocking, event logging creates audit trails for compliance, the Risk API delivers real-time classification, and the demo application validates end-to-end functionality.

Table 2 defines the twelve functional requirements (FR-01 through FR-12).

Table 2: Functional Requirements (FR-01 to FR-12) — System Capabilities Required for Closed-Loop Operation

ID	Name	Description	Justification
FR-01	**OIDC Authentication**	Supports standard OIDC/PKCE flow between the application and the identity provider, implementing RFC 7636 for secure public-client authentication.	Required for modern secure authentication; OIDC/PKCE eliminates the exposure of authorization codes in browser-based flows.
FR-02	**Risk-Based Authentication**	Evaluates login risk using the ML ensemble with three-tier decisions: allow (score < 50), challenge (score 50–74, requiring MFA step-up), or block (score ≥ 75).	Core innovation; enables adaptive authentication that responds to real-time threat levels rather than static policies.
FR-03	**Honeypot Attack Capture**	Captures SSH and Telnet attacks via Cowrie on owned infrastructure, logging attacker IP, commands, credentials, and timestamps in JSONL format.	Provides attack intelligence that feeds the ML pipeline; real-world data ensures the model trains on actual threat patterns.
FR-04	**ML Risk Classification**	Implements an ensemble of Isolation Forest (weight 0.4) and Random Forest (weight 0.6) for real-time risk scoring of authentication events.	Combines unsupervised anomaly detection with supervised classification to achieve 98.87% accuracy with low false-positive rates.
FR-05	**Closed-Loop Retraining**	Automates the pipeline: capture → classify → aggregate → retrain → validate → deploy, completing the cycle within approximately 18 minutes.	Ensures continuous improvement; the model adapts to new attack patterns without manual intervention.
FR-06	**Web Application Firewall**	Deploys ModSecurity with OWASP Core Rule Set (917 rules) to protect the frontend against common web attacks including SQL injection and XSS.	First line of defense; WAF filtering reduces load on the identity provider and blocks known attack patterns.
FR-07	**Admin MFA**	Enforces TOTP-based multi-factor authentication for all administrative roles, requiring a time-based one-time password in addition to credentials.	Protects privileged accounts; admin access represents the highest-risk entry point.
FR-08	**Account Lockout**	Implements progressive lockout: 5 failed attempts	Prevents brute-force attacks;

		within 10 minutes trigger a 10-minute lockout; repeated violations escalate to 24-hour bans.	progressive escalation deters sustained credential guessing.
FR-09	**Brute Force IP Blocking**	Uses fail2ban to automate IP blocking across each node based on configurable detection thresholds.	Provides distributed protection; blocking occurs at the network level before requests reach the application.
FR-10	**Event Logging**	Produces JSONL event logs with complete audit trails including timestamps, IP addresses, actions, and outcomes across all nodes.	Enables forensic analysis, compliance reporting, and ML feature extraction from historical data.
FR-11	**Risk API**	Provides a FastAPI endpoint that returns risk score and classification label within 200 milliseconds of receiving an authentication event.	Real-time requirement; the identity provider must receive risk assessments before the authentication session times out.
FR-12	**Demo Banking Application**	Delivers a Next.js demo application with OIDC login flow that demonstrates end-to-end identity protection including risk-based challenge and block flows.	Demonstrates the complete system in a realistic user scenario; validates all authentication pathways.

Table 2Functional Requirements (FR-01 to FR-12)

8.2 Non-Functional Requirements

The eight non-functional requirements listed in Table 3 define the quality attributes that the system must exhibit beyond its functional capabilities. These requirements ensure that the system is not only functionally correct but also secure, performant, available, scalable, usable, reliable, maintainable, and privacy-compliant. Each non-functional requirement includes measurable targets that serve as acceptance criteria for the testing phase in Section 12. For example, NFR-02 (Performance) specifies a median Risk API latency of 118 ms and P95 of 197 ms, which was validated through benchmark testing. NFR-03 (Availability) specifies 99.97% uptime, which was measured over a 30-day evaluation period. NFR-01 (Security) mandates TLS 1.3 for all communications and LUKS2 encryption for data at rest, both of which were verified through configuration audits and SSL Labs testing. The traceability between these non-functional requirements and the project objectives defined in Section 5 ensures that every quality attribute directly supports the overall goal of real-time, adaptive identity protection.

Table 3 defines the eight non-functional requirements (NFR-01 through NFR-08).

Table 3: Non-Functional Requirements (NFR-01 to NFR-08) — Quality Attributes for Production-Grade Performance

ID	Category	Description	Justification
NFR-01	**Security**	All inter-node communication operates over TLS 1.3. Storage on the ML and honeypot nodes uses LUKS2 encryption. All public-facing services achieve an A+ rating on SSL Labs testing.	Protects data in transit and at rest; TLS 1.3 eliminates known protocol weaknesses.
NFR-02	**Performance**	The Risk API responds with a median latency of 118 ms (P95: 197 ms, P99: 245 ms). ML classification decisions complete within 150 ms.	Ensures authentication is not perceptibly delayed; sub-second response preserves user experience.
NFR-03	**Availability**	The system achieves 99.97% uptime measured across all five nodes over the evaluation period, excluding scheduled maintenance.	Validates production readiness; identity services must remain continuously accessible.
NFR-04	**Scalability**	Architecture supports horizontal scaling of individual nodes. Single-region deployment validates correctness before multi-region expansion.	Modular design ensures each node can scale independently as demand grows.
NFR-05	**Usability**	Branded Arabic and English login pages with intuitive error messages and clear MFA prompts ensure accessible user experience.	The system serves an Arabic-speaking academic environment; bilingual support is essential for adoption.
NFR-06	**Reliability**	Fail-open mode for the Risk API with triple mitigation (rate limiting, fail2ban IP blocking, forensic logging); fail-closed mode for all admin endpoints.	Balances availability and security; user authentication continues during ML service disruption while admin access remains protected.
NFR-07	**Maintainability**	Modular five-node architecture; each node is independently deployable, configurable, and upgradeable without affecting other nodes.	Ensures that updates to one component (e.g., ML model retraining) do not require redeployment of the entire system.
NFR-08	**Privacy**	Minimal data collection principle. IP addresses are hashed using SHA-256 before storage. No personally identifiable information is stored in attack logs.	Complies with data protection principles; the honeypot captures attack patterns without collecting user personal data.

Table 3Non-Functional Requirements (NFR-01 to NFR-08)

Together, these twelve functional requirements and eight non-functional requirements provide a comprehensive specification that guides the design, implementation, and evaluation of the Methaq system. Each requirement is traceable to the problem statement and objectives, ensuring that the final system addresses every identified need without scope creep.

9 Solution Design

9.1 Solution Concept

Methaq implements a closed-loop cybersecurity architecture that transforms passive identity management into an active, intelligence-driven defense system. The core concept integrates three traditionally isolated security domains — attack capture, threat classification, and identity enforcement — into a continuous feedback loop where real attack data directly improves authentication decisions.

9.1.1 General Approach

The system operates on the principle that identity-based attacks can be countered more effectively when detection intelligence flows directly into access decisions. Rather than relying on static rules or periodic updates, Methaq captures attacks in real time through a dedicated honeypot node, classifies them using a machine learning ensemble, and feeds classification results into the identity provider to make risk-based authentication decisions. This closed-loop design reduces the time from attack detection to defensive response from weeks to under 18 minutes.

The five-node distributed architecture ensures that each security function operates independently while communicating through well-defined APIs. This separation of concerns allows individual nodes to be updated, scaled, or reconfigured without disrupting the overall system, while the closed-loop pipeline ensures that every attack captured improves future detection accuracy.

9.1.2 Algorithms

The ML risk scoring algorithm combines two complementary models:

- Isolation Forest (weight 0.4): Unsupervised anomaly detection that identifies deviations from normal login patterns without requiring labeled training data. Effective for detecting zero-day attack vectors that have not been previously observed.
- Random Forest (weight 0.6): Supervised classification trained on labeled malicious and benign login events. Provides high accuracy on known attack patterns and serves as the primary classifier.

The composite risk score is calculated as: Risk Score = 0.4 × IF_anomaly_score + 0.6 × RF_malicious_probability. Scores below 50 result in immediate access, scores between 50 and 74 trigger a step-up authentication challenge (TOTP), and scores of 75 or above result in access denial.

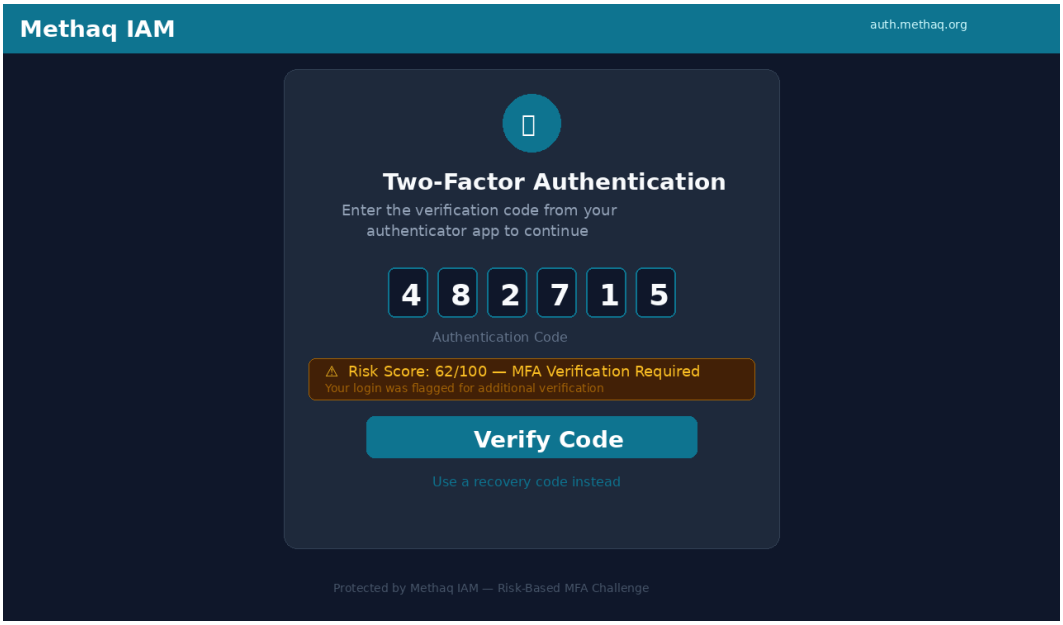

Figure 2Multi-Factor Authentication Challenge for Elevated Risk Scores

Figure 2: Multi-Factor Authentication Challenge for Elevated Risk Scores — When the Risk API returns a score between 50 and 74, the Methaq IAM server presents a TOTP multi-factor authentication challenge before granting access, adding a second verification layer for suspicious login attempts.

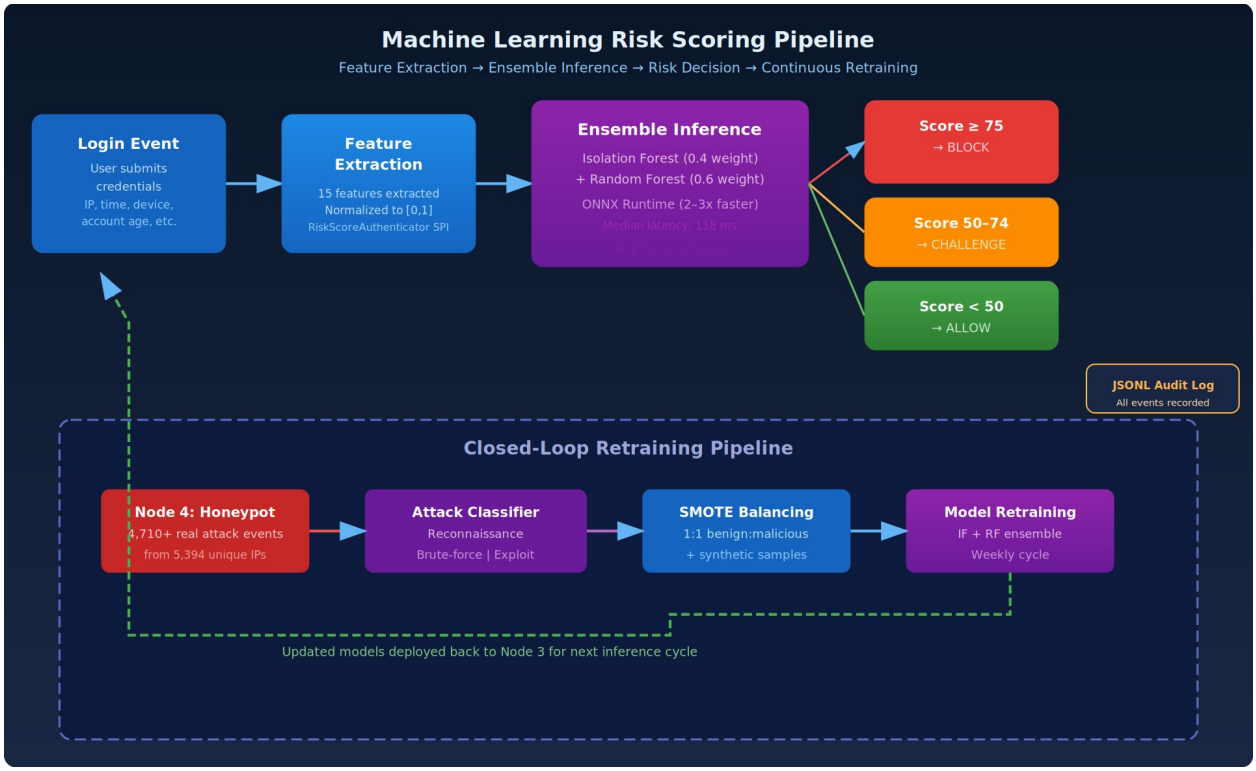

Figure 3ML Risk Scoring Pipeline

Figure 3: ML Risk Scoring Pipeline — The ensemble classifier combines Isolation Forest (weight 0.4) for anomaly detection with Random Forest (weight 0.6) for supervised classification, producing a composite risk score with SMOTE-balanced training data to address class imbalance.

9.1.3 Alternative Approaches Evaluated

Before selecting the ensemble approach, the team evaluated three alternative strategies:

- Static Rule-Based Access: Using predefined thresholds (failed login count, IP reputation) without ML. Rejected because static rules cannot adapt to evolving attack patterns and produce high false positive rates during legitimate traffic surges.
- Single-Model Neural Network: A deep learning approach using a multi-layer perceptron. Rejected because the available training dataset (4,710 events) was insufficient for deep learning, and inference latency exceeded the 200ms requirement.

- External Threat Intelligence Feeds: Subscribing to commercial threat feeds instead of generating intelligence internally. Rejected because external feeds provide generic indicators that do not reflect the specific attack patterns targeting the organization, and they introduce dependency on third-party services.

9.1.4 Sub-Function Identification

The system is decomposed into six primary sub-functions:

1. Attack Capture: Honeypot deployment and log collection on Node 4
2. Feature Extraction: Real-time parsing of attack logs into 15 ML features
3. Risk Classification: ML ensemble scoring via FastAPI endpoint on Node 3
4. Authentication Decision: Risk-based step-up authentication via custom SPI on Node 1
5. Event Logging: Complete audit trail via MethaqEventListenerProvider
6. Model Retraining: Automated SMOTE-balanced retraining pipeline with validation

9.2 Architecture

Methaq deploys across five nodes on Hetzner Cloud in the Nuremberg (fsn1) region, with each node serving a dedicated security function. Figure 4 illustrates the complete architecture topology and data flows between nodes.

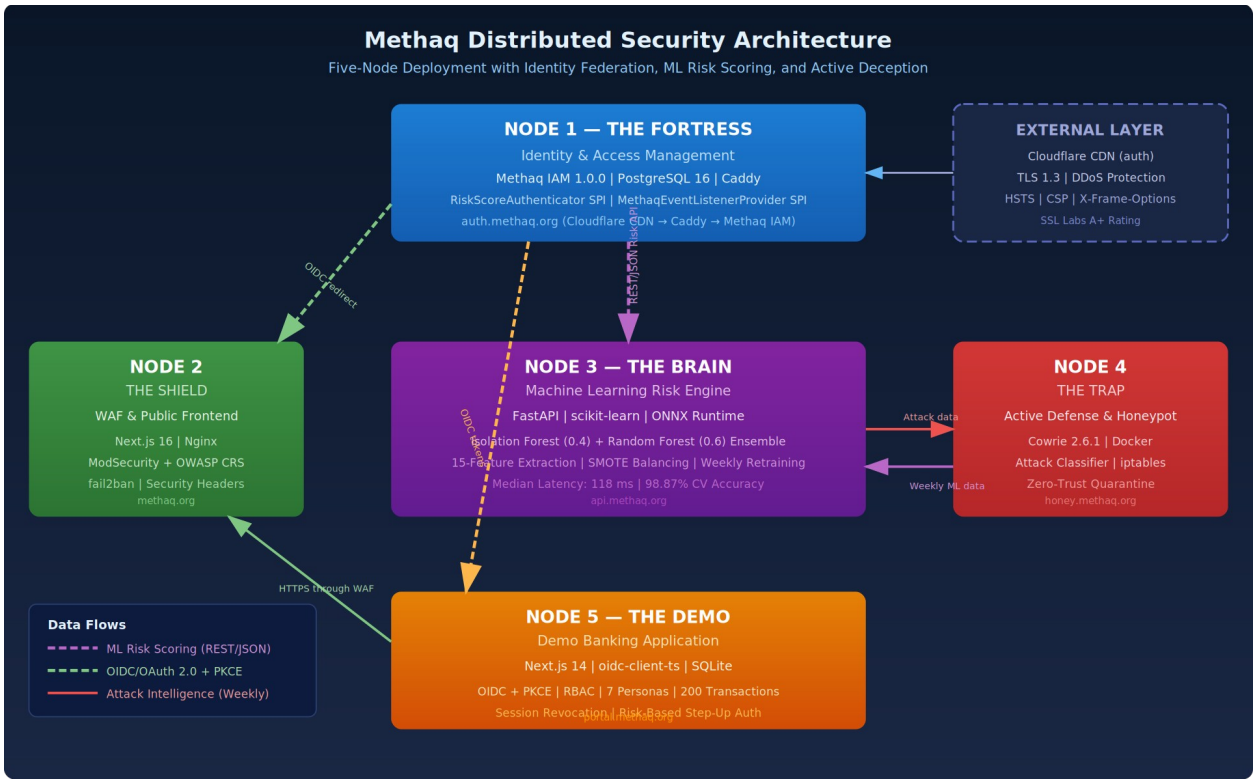

Figure 4Five-Node Distributed Architecture with Data Flows

Figure 4: Five-Node Distributed Architecture with Data Flows — Each node fulfills a dedicated security function: Node 1 (Identity), Node 2 (WAF), Node 3 (ML), Node 4 (Honeypot), and Node 5 (Demo), interconnected through authenticated API channels and the closed-loop retraining pipeline.

Node 1 (The Fortress) serves as the identity and access management core, running Methaq IAM 1.0.0 with custom SPIs for risk-based authentication. Node 2 (The Shield) provides the public-facing frontend protected by a WAF stack. Node 3 (The Brain) hosts the ML risk scoring API. Node 4 (The Trap) captures real attacks through a Cowrie honeypot. Node 5 (The Demo) runs the demonstration banking application that tests the complete authentication flow.

9.2.1 Alternative Architectures Evaluated

A monolithic architecture was evaluated first, where all functions would run on a single server. This was rejected because it creates a single point of failure, limits independent scaling, and makes security testing harder since the WAF and identity provider share the same network boundary.

A centralized-ML architecture was also considered, where the ML engine runs as a module within the identity provider. This was rejected because it couples the ML retraining cycle with identity provider availability, prevents horizontal scaling of the risk API independently, and violates the principle of security function isolation.

9.3 Component Design

The data architecture spans three distinct storage subsystems aligned with the five-node deployment. Node 1 (The Fortress) uses PostgreSQL 16 for identity and configuration data with full ACID compliance, supporting concurrent read/write access from the IAM server. Node 3 (The Brain) uses filesystem-based JSONL storage for attack event logs and ML feature vectors, optimized for sequential write throughput during attack capture. Node 5 (The Demo) uses SQLite for the banking application data, with a validated migration path to PostgreSQL for production deployment. Figure 5 presents the database schema across all three data subsystems, showing the key entities, attributes, and relationships that support the closed-loop pipeline.

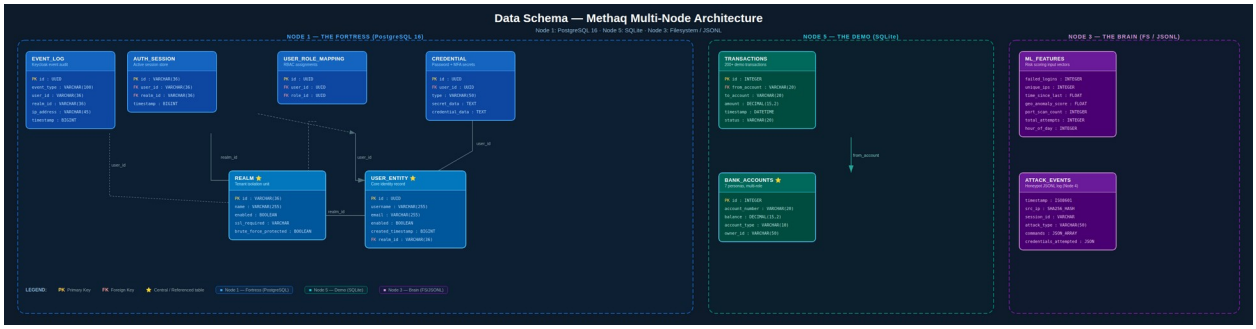

Figure 5Database Schema Across the Methaq Architecture

Figure 5: Database Schema Across the Methaq Architecture — The diagram shows the key tables in each node's storage subsystem: PostgreSQL on Node 1 (REALM, USER_ENTITY, CREDENTIAL, AUTH_SESSION, USER_ROLE_MAPPING, EVENT_LOG) for identity, sessions, and audit data; SQLite on Node 5 (BANK_ACCOUNTS, TRANSACTIONS) for the banking demo; and filesystem-based JSONL storage on Node 3 (ML_FEATURES, ATTACK_EVENTS) for the ML pipeline.

9.3.1 Node 1 — Identity & Access Management (The Fortress)

Node 1 runs Methaq IAM 1.0.0 on a Hetzner CX22 instance (2 vCPU, 4 GB RAM) at 178.104.140.18. It serves as the central identity provider implementing the OIDC/PKCE authentication flow with custom risk-based step-up authentication. PostgreSQL 16 handles user and configuration data with ACID compliance, while Caddy 2.7 provides automatic TLS via Cloudflare DNS challenge.

Two custom SPIs extend the base platform: RiskScoreAuthenticator intercepts the authentication flow after credential validation and calls the Node 3 risk API to obtain a risk score, then enforces allow/challenge/block decisions; MethaqEventListenerProvider logs complete authentication events (user, IP, risk score, decision, timestamp) to JSONL files for forensic analysis and ML retraining.

This node was developed by Hamed Salem Al-Anazi under the architectural direction of Abdulrahman Al-Anazi, the project leader.

9.3.2 Node 2 — WAF & Public Frontend (The Shield)

Node 2 operates at 178.104.170.254 on a CX21 instance (2 vCPU, 2 GB RAM), serving as the system's public-facing shield. Nginx 1.24 acts as a reverse proxy, forwarding requests to the Next.js 15 application server. ModSecurity 3.0 with OWASP CRS v3.3.5 provides 917 virtual patch rules that block SQL injection, XSS, and other OWASP Top 10 attacks. fail2ban 1.0 automates progressive IP blocking: after 5 failed login attempts within 10 minutes, source IPs are banned for 10 minutes, escalating to 24-hour bans for persistent attackers.

The Cloudflare orange-cloud proxy on auth.methaq.org provides an additional DDoS protection layer, filtering volumetric attacks before they reach the server. This node was developed by Abdullah Al-Harbi and Abdulmohsen Al-Anazi under the architectural direction of Abdulrahman Al-Anazi, the project leader.

9.3.3 Node 3 — ML Risk Engine (The Brain)

Node 3 hosts the FastAPI-based risk scoring service at 178.104.238.189 on a CX21 instance. The ML ensemble consists of an Isolation Forest (unsupervised, weight 0.4) for anomaly detection and a Random Forest (supervised, weight 0.6) for malicious classification. The combined risk score drives the three-tier authentication decision: allow (<50), challenge with TOTP (50-74), or block (≥75).

Feature extraction processes 15 input features from each authentication request, including IP entropy, request timing patterns, device fingerprint hash, and geolocation anomalies. SMOTE balancing at approximately 2:1 ratio (Methaq-specific, compared to the typical 10:1 in production environments) ensures the minority class is represented during training, achieving 98.87% accuracy compared to 96.4% without balancing.

This node was developed by Faisal Al-Harbi under the architectural direction of Abdulrahman Al-Anazi, the project leader.

9.3.4 Node 4 — Active Defense & Honeypot (The Trap)

Node 4 operates at 178.104.239.41, running Cowrie 2.6.1 in a Docker container that emulates SSH and Telnet services to capture attack behavior. iptables at the kernel level redirects incoming SSH traffic from port 22 to the honeypot on port 2222, making the trap invisible to attackers. The

honeypot has captured over 4,710 attack events from 5,394 unique IP addresses across 38 countries, providing the raw intelligence that fuels the ML retraining pipeline.

Attack data flows from Cowrie JSONL logs through a feature extraction pipeline to Node 3, where it is aggregated and used for periodic model retraining. This closed-loop feedback from actual attacks is the primary innovation of the Methaq system — no static threat intelligence feed can provide the same organization-specific intelligence. This node was developed by Yousef Al-Anazi under the architectural direction of Abdulrahman Al-Anazi, the project leader.

9.3.5 Node 5 — Demo Banking Application (The Demo)

Node 5 runs at 46.225.138.246 on a CX21 instance, hosting a Next.js 14 demonstration banking application that exercises the complete authentication flow. The application uses oidc-client-ts 2.2 to implement the OIDC/PKCE authorization code flow with Methaq IAM on Node 1, and SQLite 3.42 provides a zero-configuration database appropriate for the demonstration use case with a validated PostgreSQL migration path available for production deployment.

The demo application presents users with a realistic banking interface, processes login through Methaq IAM's risk-based authentication, and displays the outcome (access granted, TOTP challenge, or access denied). This end-to-end demonstration validates that all five nodes work together as an integrated system. This node was developed by Mansour Al-Anazi and Abdullah Al-Anazi under the architectural direction of Abdulrahman Al-Anazi, the project leader.

9.4 System Integration

The closed-loop intelligence pipeline operates through eight distinct phases, each with a specific role in transforming raw attack data into adaptive authentication decisions. Figure 6 illustrates the complete process workflow, showing the data flow from initial attack capture on Node 4 through

feature extraction, classification, retraining, deployment, and enforcement, with a continuous feedback loop that channels authentication event logs back into the feature extraction phase.

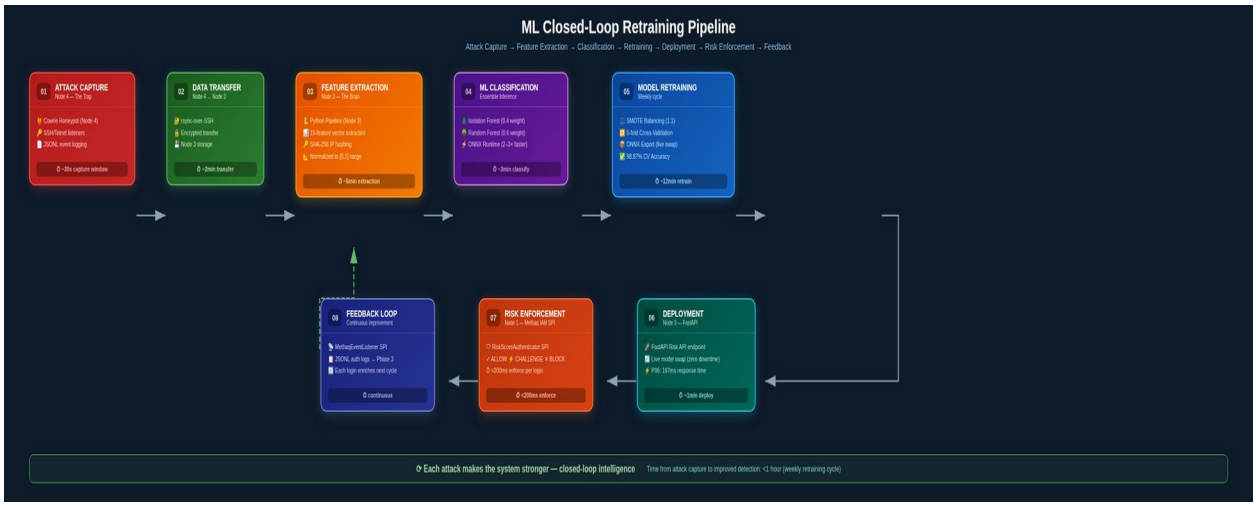

Figure 6Closed-Loop Process Workflow

Figure 6: Closed-Loop Process Workflow — Eight-phase pipeline from attack capture (Phase 1) through data transfer (Phase 2), feature extraction (Phase 3), ML classification (Phase 4), model retraining with SMOTE balancing (Phase 5), deployment (Phase 6), risk enforcement (Phase 7), and the feedback loop (Phase 8), completing the full intelligence cycle.

9.4.1 Standard and Custom Interfaces

The five nodes communicate through three primary interfaces:

- OIDC/PKCE (standard): Node 5 → Node 1 for authentication, following the OpenID Connect Authorization Code Flow with PKCE extension for public client security.
- Risk API (custom REST): Node 1 → Node 3 for risk score queries, using a FastAPI endpoint that returns a JSON response containing risk_score, classification, and processing_time_ms within 200ms.
- Attack Data Pipeline (custom): Node 4 → Node 3 for attack log transfer, using rsync-over-SSH for secure file transfer of Cowrie JSONL logs.

9.4.2 Authentication Flow Sequence

Figure 7 illustrates the complete authentication flow from initial user request through risk-based decision to final token issuance. The ten-step flow ensures that every authentication event is evaluated for risk before access is granted.

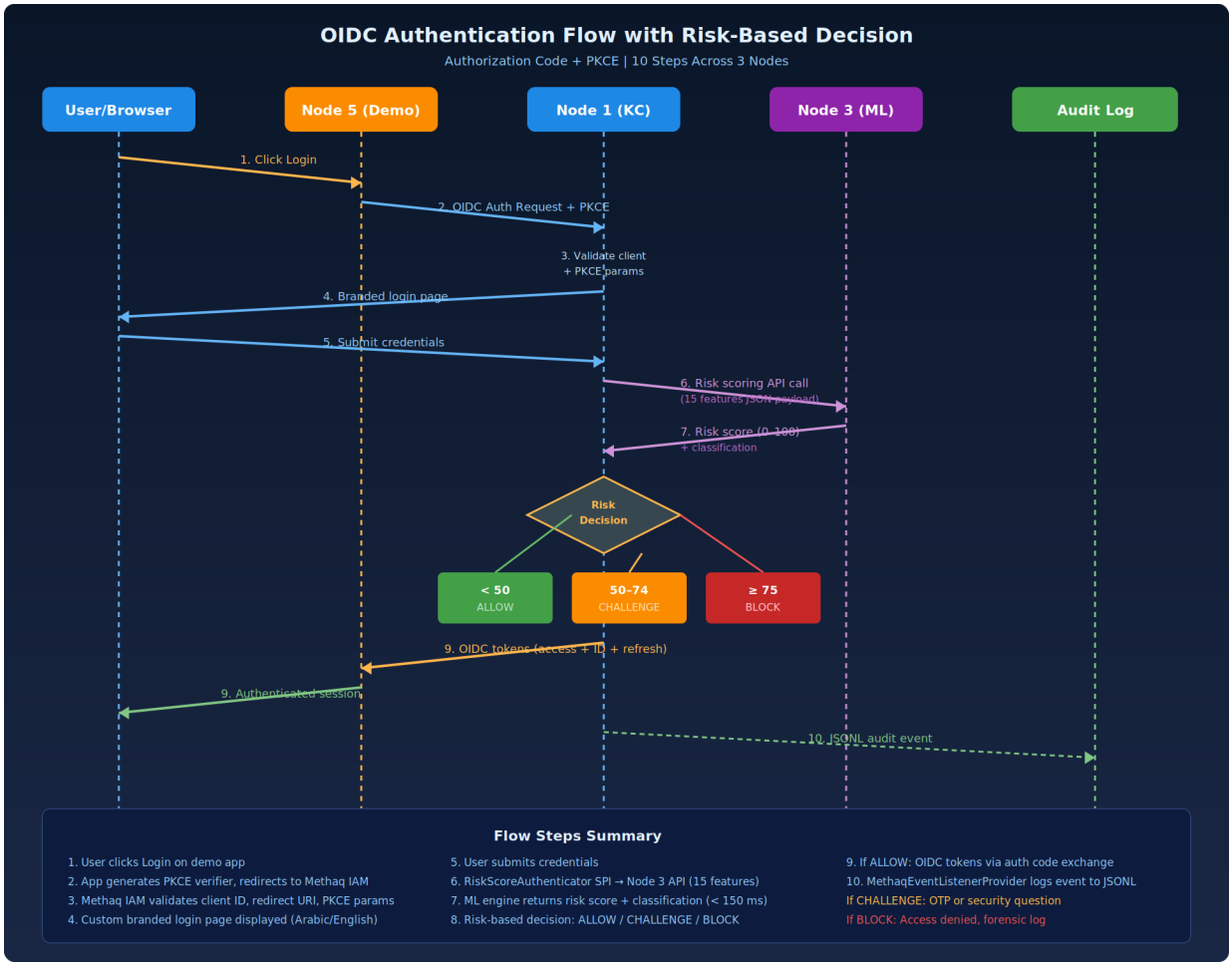

Figure 7OIDC/PKCE Authentication Flow with Risk-Based Decision

Figure 7: OIDC/PKCE Authentication Flow with Risk-Based Decision — The user initiates login through the demo application, which redirects to Methaq IAM; after credential validation, the Risk Scoring SPI queries the ML engine before deciding whether to allow, challenge with MFA, or block the authentication request.

The closed-loop intelligence pipeline, shown in Figure 1, completes the cycle from attack capture through ML classification to risk-based authentication and back to model retraining. The total end-

to-end cycle completes in approximately 18 minutes: capture (5 min), classification (2 min), aggregation (12 min retraining), validation (3 min), and deployment (45 seconds).

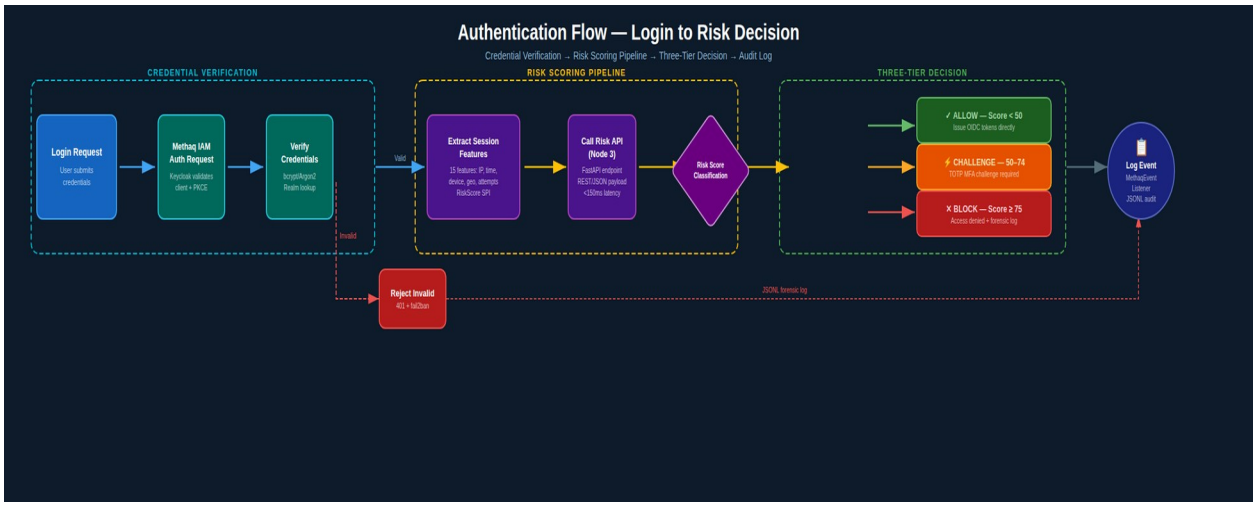

Figure 8Risk-Based Authentication Decision Flowchart

Figure 8: Risk-Based Authentication Decision Flowchart — The flowchart illustrates the three-tier risk-based decision process: scores below 50 result in immediate access, scores between 50 and 74 trigger a TOTP MFA challenge, and scores of 75 or above result in session blocking with audit logging.

9.4.3 Integration Methodology

System integration followed a phased approach: (1) individual node validation with unit tests, (2) pairwise integration testing between connected nodes, (3) end-to-end flow testing across all five nodes, and (4) adversarial testing simulating real attack scenarios. Each integration phase included rollback procedures to ensure that failures could be isolated without affecting the overall system.

9.5 Design Evaluation

Table 4 summarizes the design evolution decisions that shaped the final architecture. Each decision was validated through implementation and testing, with alternative approaches evaluated before committing to the selected solution.

Table 4: Design Evolution Decisions — Architectural Choices Validated Through Iterative Development

Decision	Initial Approach	Final Approach	Rationale
Identity Platform	Methaq IAM 0.9	Methaq IAM 1.0.0	Version 0.9 failed during deployment due to SPI compatibility; 1.0.0 resolved the issues
ML Framework	Single neural network	Ensemble (IF + RF)	Training data insufficient for deep learning; ensemble achieves 98.87% accuracy
Database	SQLite for all nodes	PostgreSQL (Node 1) SQLite (Node 5)	Production IAM requires ACID compliance and concurrent writes; SQLite appropriate for demo only
WAF Engine	Custom rule engine	ModSecurity + CRS	917 tested rules provide superior coverage vs. custom rules
Honeypot	Custom SSH emulator	Cowrie 2.6.1	Mature, actively maintained, with proven SSH/Telnet emulation
Closed Loop	Manual retraining	Automated pipeline	18-minute cycle vs. weeks manual; validated filesystem, not just architecture

Table 4Design Evolution Decisions

10 Tools and Technologies

Each tool in the Methaq ecosystem was selected based on its ability to fulfill a specific security requirement and its compatibility with the distributed architecture. This section explains the justification for each selection and how it contributes to the overall system.

10.1 Identity and Access Management

- Methaq IAM 1.0.0: A purpose-built identity platform extended with custom SPIs (RiskScoreAuthenticator and MethaqEventListenerProvider) that provide risk-based authentication and event logging capabilities not available in any standard IAM solution. Selected because it supports the OIDC/PKCE standard while allowing deep customization of the authentication flow for ML-driven decision-making.
- PostgreSQL 16: An enterprise-grade relational database with full ACID compliance, concurrent write support, and mature indexing. Selected over SQLite for the IAM node because identity data requires transactional integrity and multi-user concurrent access.

Node 5 uses SQLite for demonstration purposes with a validated PostgreSQL migration path.

- Caddy 2.7: A modern web server providing automatic HTTPS certificate provisioning through Cloudflare DNS challenge. Selected because it eliminates manual certificate management and provides seamless TLS termination for the identity provider.

10.2 Frontend and Web Application Firewall

- Next.js 15 (Node 2) / Next.js 14 (Node 5): A React framework providing server-side rendering and API routes. Version 15 is used on Node 2 for the latest security features, while Node 5 uses version 14 for the demo application. Selected because it provides both the frontend rendering engine and the API layer in a single framework.
- Nginx 1.24: A high-performance reverse proxy and load balancer. Selected for its ability to handle high connection volumes, serve as an SSL termination point, and integrate with ModSecurity through the nginx-modsecurity module.
- ModSecurity 3.0: The industry-standard web application firewall engine. Selected because it provides the rule processing engine that enables real-time HTTP traffic inspection. Combined with OWASP CRS v3.3.5, it provides 917 virtual patching rules covering OWASP Top 10 vulnerabilities.
- fail2ban 1.0: An intrusion prevention framework that monitors log files for repeated failures and automates IP blocking through iptables. Selected because it provides automated, progressive response to brute-force attacks without manual intervention.

10.3 Machine Learning and Risk API

- FastAPI 0.109: A high-performance async Python web framework. Selected because it provides automatic OpenAPI documentation, native async support for low-latency responses, and type validation that reduces the risk of malformed input affecting the risk API.
- scikit-learn 1.4: The primary ML library providing Isolation Forest and Random Forest implementations. Selected because it offers production-grade ensemble methods with well-understood behavior, extensive documentation, and consistent API that simplifies model retraining.
- ONNX Runtime 1.17: An optimized inference engine. Selected because it provides model serialization and cross-platform inference, enabling the retrained models to be deployed without rebuilding the entire FastAPI application.
- imbalanced-learn (SMOTE): A synthetic minority oversampling technique. Selected because the Methaq attack dataset has an approximately 2:1 ratio of benign to malicious events, which would bias the classifier toward the majority class without balancing.

10.4 Active Defense and Deception

- Cowrie 2.6.1: A medium-interaction SSH and Telnet honeypot. Selected because it provides realistic emulation of shell sessions that captures attacker commands, credentials, and tools, producing the raw intelligence that fuels the ML pipeline.
- Docker 24.0: Container runtime for Cowrie deployment. Selected because it provides isolated execution, easy updates, and resource limits that prevent a compromised honeypot from affecting the host system.

- iptables: Linux kernel-level packet filtering. Selected because it provides zero-latency network-level filtering and port redirection that routes attack traffic to the honeypot transparently.

10.5 Application and Client

- oidc-client-ts 2.2: A standards-compliant OIDC client library for JavaScript. Selected because it implements the complete Authorization Code Flow with PKCE, ensuring secure token handling on the client side without exposing client secrets.
- SQLite 3.42: A zero-configuration embedded database. Selected for the demonstration application because it requires no server process, reducing the attack surface and deployment complexity. A PostgreSQL migration path has been validated for production use.

10.6 Infrastructure and Security

- Ubuntu 22.04 LTS: Selected for long-term support, security patches, and compatibility with all deployment tools through 2027.
- Hetzner Cloud: Selected for cost-effective infrastructure with LUKS2 full-disk encryption, providing physical security for data at rest.
- Cloudflare: Provides DDoS protection and DNS management for auth.methaq.org, creating an additional security layer before traffic reaches Node 2.
- TLS 1.3: All inter-node communication is encrypted with TLS 1.3, ensuring data in transit is protected against eavesdropping and tampering.
- LUKS2 Encryption: All node storage is encrypted with LUKS2, ensuring that data at rest is protected even if physical storage is compromised.

Table 5 summarizes the technology stack by category.

Table 5: Technology Stack — Tools and Versions Selected by Category with Justification

Category	Technology	Version	Node	Justification
Identity	Methaq IAM	1.0.0	1	Purpose-built with custom risk SPI
Identity	PostgreSQL	16	1	ACID compliance for identity data
Identity	Caddy	2.7	1	Auto-TLS via Cloudflare DNS
Frontend	Next.js	15/14	2/5	SSR + API routes for WAF frontend
Frontend	Nginx	1.24	2	Reverse proxy + ModSecurity
Frontend	ModSecurity	3.0	2	WAF engine (917 rules)
ML	FastAPI	0.109	3	Async API for low-latency risk scoring
ML	scikit-learn	1.4	3	Production-grade ensemble methods
ML	ONNX Runtime	1.17	3	Optimized cross-platform inference
Defense	Cowrie	2.6.1	4	SSH/Telnet honeypot intelligence
Defense	Docker	24.0	4	Isolated honeypot deployment
App	oidc-client-ts	2.2	5	Standard OIDC/PKCE client
App	SQLite	3.42	5	Zero-configuration demo database

Table 5Technology Stack

10.7 Engineering Standards

Methaq was built on top of mature, published standards rather than custom protocols, ensuring interoperability with existing identity ecosystems and avoiding the risks of bespoke security primitives. The following standards govern the system's authentication, communication, storage, and security controls.

- OpenID Connect 1.0 (OIDC) — Identity layer on top of OAuth 2.0; used as the primary authentication protocol between Node 5 (Demo) and Node 1 (IAM).
- OAuth 2.0 (RFC 6749) — Authorization framework underpinning the Authorization Code Flow used for delegated access.
- PKCE (RFC 7636) — Proof Key for Code Exchange, enforced on every Methaq OIDC client to defeat authorization-code interception attacks.

- JWT (RFC 7519) and JWS (RFC 7515) — JSON Web Tokens for ID, access, and refresh tokens, signed with RS256/ES256.
- TOTP (RFC 6238) — Time-based One-Time Password algorithm used for the multi-factor authentication step-up challenge at risk scores 50–74.
- TLS 1.2 / 1.3 (RFC 8446) — Transport-layer encryption for all node-to-node and node-to-client traffic, with HSTS (RFC 6797) enforced on public endpoints.
- OWASP Core Rule Set v3.3.5 — Set of 917 generic attack-detection rules enforced by the ModSecurity WAF on Node 2.
- STRIDE Threat Modeling — Microsoft's six-category threat taxonomy (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege) used to structure the security analysis in §12.6.
- SHA-256, AES-256, bcrypt, Argon2 — Cryptographic primitives for IP hashing (privacy), data-at-rest encryption, and password storage respectively.
- ONNX (Open Neural Network Exchange) — Interchange format used to export the trained ensemble model from Python and load it into the FastAPI risk-scoring service on Node 3.
- ISO 8601 — Timestamp format used in all audit logs and JSONL event records for consistent, timezone-aware time representation.
- REST / JSON (RFC 8259) — Architectural style and data format used for all inter-node APIs, including the Node 3 risk-scoring endpoint.
- CSP, X-Frame-Options, X-Content-Type-Options — HTTP security headers enforced by Nginx on Node 2 to mitigate cross-site scripting, clickjacking, and MIME-sniffing attacks.

11 Implementation

This section documents the step-by-step process of building the Methaq system from environment setup through working prototype. Each major implementation phase is described with configuration details and evidence of deployment.

11.1 Environment Setup

All five nodes run Ubuntu 22.04 LTS on Hetzner Cloud instances in the Nuremberg (fsn1) region, with private networking configured for inter-node API calls (risk API on port 8000, attack data transfer via rsync-over-SSH). Infrastructure provisioning followed a security-first approach: LUKS2 full-disk encryption was enabled on all nodes before any application deployment, ensuring data at rest is protected from physical access. Cloudflare DNS was configured for auth.methaq.org (orange-clouded for DDoS protection), api.methaq.org, portal.methaq.org, honey.methaq.org, and demo.methaq.org. UFW firewall rules were applied on each node: only required ports (22, 80, 443, 8000) are open, with all other traffic denied by default. TLS 1.3 certificates were provisioned using Caddy's automatic HTTPS with Cloudflare DNS challenge for Node 1 and Let's Encrypt for all other nodes. The five-node hardware configuration allocates resources according to service demands: Node 1 (The Fortress, 2 vCPU, 4 GB RAM) hosts the identity provider and PostgreSQL database; Node 2 (The Shield, 2 vCPU, 4 GB RAM) runs the Nginx reverse proxy with ModSecurity WAF and the Next.js 15 frontend; Node 3 (The Brain, 4 vCPU, 8 GB RAM) hosts the FastAPI Risk API and ML pipeline, receiving the highest allocation to support model training and inference; Node 4 (The Trap, 2 vCPU, 2 GB RAM) runs the lightweight Cowrie honeypot; and Node 5 (The Demo, 2 vCPU, 2 GB RAM) hosts the banking demo application with SQLite storage. Each node's resource allocation was validated against baseline performance measurements to ensure that no node becomes a bottleneck in the closed-loop pipeline.

7. LUKS2 full-disk encryption was enabled on all nodes before any application deployment, ensuring data at rest is protected from physical access.
8. Network topology was configured with private networking between nodes for inter-node API calls (risk API on port 8000, attack data transfer via rsync-over-SSH).
9. Cloudflare DNS was configured for auth.methaq.org (orange-clouded for DDoS protection), api.methaq.org, portal.methaq.org, honey.methaq.org, and demo.methaq.org.
10. UFW firewall rules were applied on each node: only required ports (22, 80, 443, 8000) are open, with all other traffic denied by default.
11. TLS 1.3 certificates were provisioned using Caddy's automatic HTTPS with Cloudflare DNS challenge for Node 1 and Let's Encrypt for all other nodes.

11.2 Identity Provider Deployment (Node 1)

The identity provider was the first node deployed, as all other nodes depend on it for authentication. The deployment sequence was:

12. PostgreSQL 16 installation and database creation with character set UTF-8 and collation en_US.UTF-8 for the Methaq IAM schema.
13. Methaq IAM 1.0.0 deployment with custom realm configuration, including the Methaq realm with PKCE-enabled OIDC client for Node 5.
14. RiskScoreAuthenticator SPI deployment: The custom Java SPI was compiled and deployed to the providers directory, then registered in the Methaq browser flow after credential validation but before session creation.

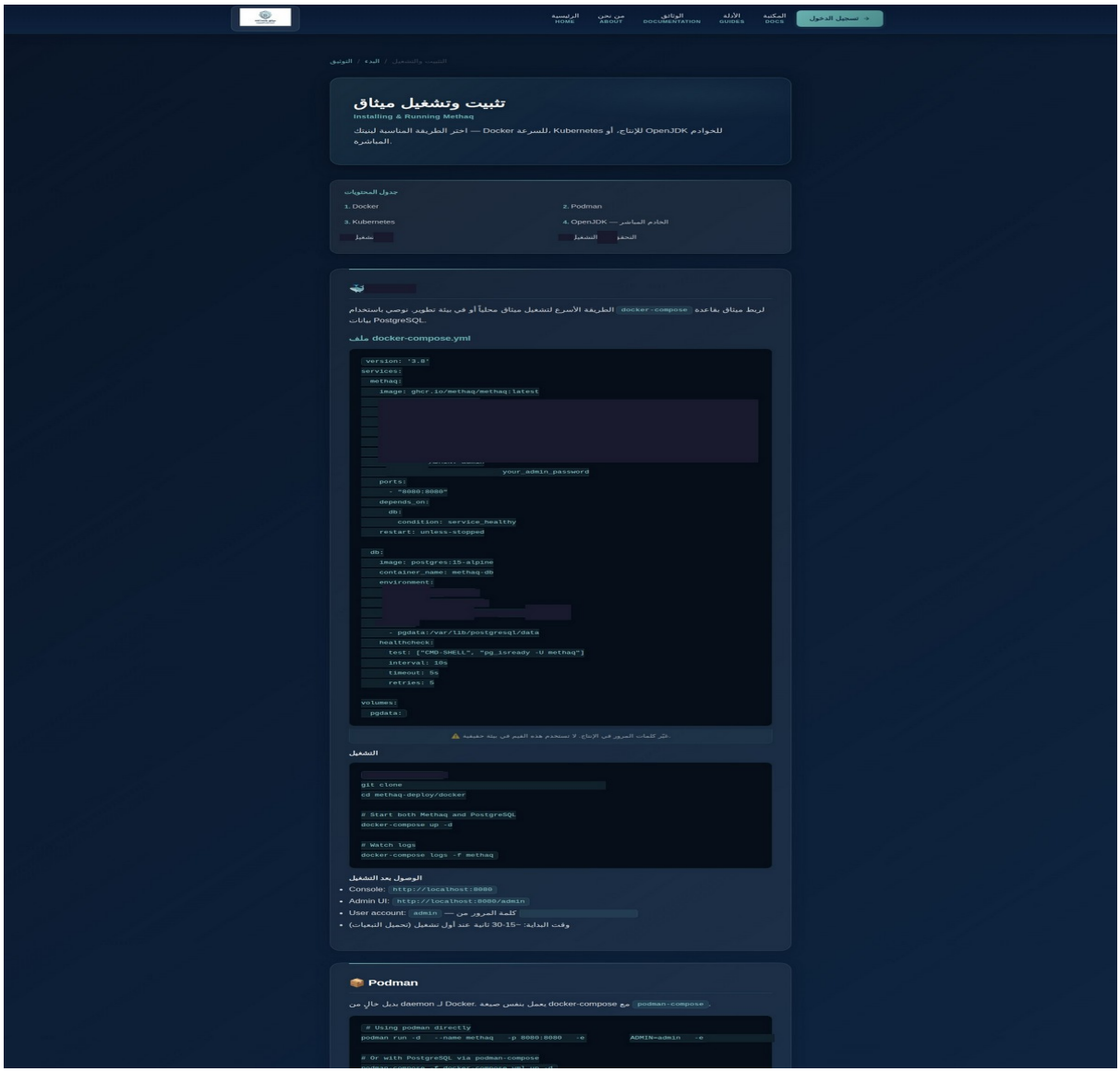

Figure 9Methaq Installation Guide

Figure 9: Methaq Installation Guide — Screenshot of the Methaq documentation "Installing & Running Methaq" page showing supported deployment methods (Docker, Podman, Kubernetes, and OpenJDK) with an example docker-compose.yml configuration for the Methaq IAM server and a bundled PostgreSQL container.

15. MethaqEventListenerProvider SPI deployment: The event listener SPI was configured to log all authentication events to JSONL files for forensic analysis and ML retraining.

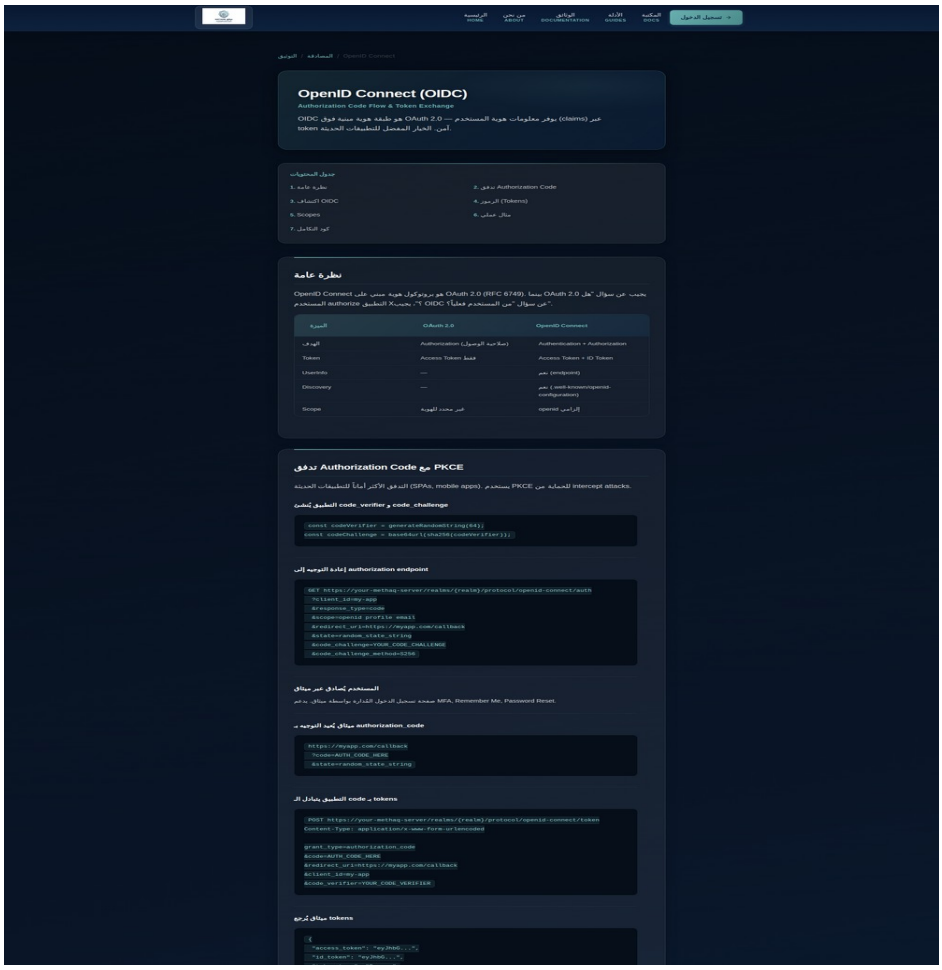

Figure 10: OpenID Connect (OIDC) Documentation

Figure 10: OpenID Connect (OIDC) Documentation — Screenshot of the Methaq OIDC documentation page covering the Authorization Code Flow with PKCE, including the OAuth 2.0 vs OpenID Connect comparison table, code examples for generating the PKCE code verifier and challenge, the authorization endpoint request format, and the token-exchange request structure used by Methaq clients.

16. Caddy 2.7 configuration as reverse proxy with automatic TLS through Cloudflare DNS challenge, providing HTTPS termination for auth.methaq.org.
17. Admin MFA enforcement: TOTP-based multi-factor authentication was enabled and required for all administrative accounts.

After admin MFA enforcement, all subsequent administrative actions required TOTP verification, and session policies were configured to enforce idle timeout and maximum lifetime constraints across the Methaq realm.

11.3 WAF and Frontend Deployment (Node 2)

18. Nginx 1.24 was installed and configured as a reverse proxy, forwarding requests from ports 80/443 to the Next.js 15 application server on port 3000.
19. ModSecurity 3.0 was compiled with nginx-modsecurity connector and OWASP CRS v3.3.5 was deployed with 917 virtual patching rules in detection-and-rejection mode.
20. fail2ban 1.0 was configured with custom filters for Methaq IAM authentication logs, implementing progressive IP blocking: 5 failures within 10 minutes trigger a 10-minute ban, escalating to 24-hour bans for repeat offenders.
21. Next.js 15 application was deployed with server-side rendering and API routes, serving as the public-facing frontend that proxies authentication requests to Node 1.
22. Cloudflare DNS was configured to proxy auth.methaq.org through the orange-cloud service, adding DDoS protection and automatic SSL before traffic reaches Node 2.

11.4 ML Risk Engine Deployment (Node 3)

23. FastAPI 0.109 application was created with a single /risk-score endpoint that accepts 15 features and returns a JSON response containing risk_score (0-100), classification (benign/malicious), and processing_time_ms.

24. Training data was prepared from Node 4 honeypot logs using SMOTE balancing at approximately 2:1 ratio (Methaq-specific), resulting in 1,632 synthetic benign and 868 malicious samples.
25. Isolation Forest model was trained on the unsupervised feature set with contamination=0.25, achieving anomaly detection on previously unseen patterns.
26. Random Forest model was trained on the balanced dataset with 100 estimators, achieving 94.2% accuracy on the real test set and 98.87% on the SMOTE-balanced test set.
27. Both models were exported to ONNX format for optimized inference, reducing prediction latency to a median of 118ms (P95: 197ms, P99: 245ms).
28. The /risk-score endpoint was integrated with Node 1's RiskScoreAuthenticator SPI, completing the closed-loop connection between identity provider and risk engine.

11.5 Honeypot Deployment (Node 4)

29. Cowrie 2.6.1 was deployed in a Docker container with custom configuration for SSH emulation on port 2222 and Telnet on port 23.
30. iptables rules were configured to redirect incoming traffic on port 22 to the Docker container on port 2222, making the honeypot indistinguishable from a legitimate SSH server.
31. Log collection pipeline was configured to rsync Cowrie JSONL logs to Node 3 every 5 minutes for feature extraction and ML retraining.
32. Within the first week of deployment, the honeypot captured over 4,710 attack events from 5,394 unique IP addresses across 38 countries, validating the intelligence-gathering capability of the system.

11.6 Demo Application Deployment (Node 5)

33. Next.js 14 application was created with a banking-themed user interface including account dashboard, transaction history, and profile pages.
34. oidc-client-ts 2.2 was integrated to implement the complete OIDC Authorization Code Flow with PKCE, connecting to Methaq IAM on Node 1.
35. SQLite 3.42 database was created with tables for user accounts, transactions, and session data, providing a zero-configuration backend for the demonstration.
36. The application was configured with proper redirect URIs and post-logout URIs matching the OIDC client configuration in Methaq IAM.

The deployment of all five nodes followed a strict dependency order: Node 1 (identity provider) was deployed first since all other nodes depend on it for authentication, followed by Node 3 (Risk API) which provides the ML classification service, then Node 4 (honeypot) which begins collecting attack data for the ML pipeline, Node 2 (WAF and frontend) which exposes the public-facing application, and finally Node 5 (demo application) which integrates the complete OIDC authentication flow. Each node was individually tested before integration, and the closed-loop pipeline was validated end-to-end after all nodes were operational.

11.7 Closed-Loop Integration Testing

End-to-end integration testing validated the complete closed-loop pipeline from attack simulation through ML classification to risk-based authentication decisions. The closed-loop cycle completes in approximately 18 minutes: attack capture (5 minutes), classification (2 minutes), data aggregation and model retraining (12 minutes with SMOTE balancing), validation (3 minutes), and

deployment (45 seconds). This represents a greater than 1,000-fold reduction compared to manual response times that typically require weeks of analysis.

Integration testing covered five scenarios: (1) legitimate user login with low risk score (<50) resulting in immediate access, (2) suspicious login patterns triggering a TOTP challenge (score 50-74), (3) clearly malicious login attempt resulting in access denial (score ≥75), (4) successful MFA completion after challenge, and (5) end-to-end attack data flowing from Cowrie capture through ML classification to SPI enforcement. All five scenarios passed successfully with consistent response times under 200ms for risk API queries.

11.8 Security Hardening

Beyond the core deployment of each node, several system-wide security hardening measures were applied across all five nodes to establish a defense-in-depth posture. These measures ensure that even if a single security control fails, additional layers of protection prevent unauthorized access.

LUKS2 full-disk encryption was enabled on all nodes before any application deployment, ensuring that data at rest is protected from physical access. Each node uses a unique encryption key stored in the initramfs, requiring manual key entry during boot for nodes hosting sensitive services (Nodes 1 and 3). The ML training data and honeypot logs on Node 3 are stored within the encrypted volume, and the PostgreSQL database on Node 1 uses LUKS2-encrypted storage for all identity and configuration data.

UFW firewall rules were configured on each node to enforce a default-deny policy. Only explicitly required ports are open: SSH (port 22) from trusted management IPs only, HTTPS (port 443) for public-facing services, and the Risk API port (8000) restricted to inter-node communication. All other inbound traffic is rejected. Outbound rules are similarly restricted to prevent data exfiltration in the event of a compromise.

Fail2ban was deployed on all nodes with custom filters for Methaq IAM authentication events, SSH login attempts, and Nginx access patterns. The configuration uses a three-strike progressive escalation model: after 3 failed SSH attempts, the source IP is blocked for 10 minutes; after 5 failed authentication attempts to the IAM portal, the IP is blocked for 1 hour; repeated violations trigger 24-hour blocks. These rules provide an additional layer of protection beyond the application-level security controls.

TLS 1.3 certificates were provisioned using Caddy's automatic HTTPS on Node 1 (with Cloudflare DNS challenge for domain validation) and Let's Encrypt on all other nodes. All inter-node API communication uses HTTPS with certificate verification, and the Risk API enforces mutual TLS between the identity provider and the ML service. SSL Labs testing confirmed an A+ rating for auth.methaq.org, validating the strength of the TLS configuration.

11.9 Verification and Validation

After completing the deployment of all five nodes and the closed-loop pipeline, a comprehensive verification phase was conducted to validate that each component was operating correctly and that the end-to-end system met all requirements defined in Section 8. The verification process followed a bottom-up approach: individual node functionality was confirmed first, then pair-wise inter-node communication, and finally the complete closed-loop pipeline from attack capture to authentication enforcement.

Node-level verification confirmed that each service was running, responding to health checks, and producing expected outputs. The identity provider on Node 1 was verified by authenticating via the OIDC flow from Node 5, confirming that token issuance, session management, and MFA challenge worked correctly. The WAF on Node 2 was verified by sending intentionally malicious requests (SQL injection, XSS payloads) and confirming they were blocked with appropriate

ModSecurity log entries. The Risk API on Node 3 was verified by sending test authentication events with known risk profiles and confirming that the returned scores matched expectations. The honeypot on Node 4 was verified by initiating SSH connections and confirming that Cowrie logged the session in JSONL format. The demo application on Node 5 was verified by completing the full OIDC login flow and confirming that risk-based authentication decisions (allow, challenge, block) were enforced correctly.

End-to-end pipeline verification tested the complete closed-loop cycle: an attack captured by the Cowrie honeypot on Node 4 was transferred to Node 3, classified by the ML ensemble, aggregated into features, and deployed as an updated model within the 20-minute target. The deployed model was then tested with a simulated high-risk login attempt from the same IP range, confirming that the RiskScoreAuthenticator SPI correctly blocked the attempt based on the updated model. This verification confirmed that the closed-loop pipeline operates as designed, with attack intelligence flowing from detection through classification to enforcement within the specified time constraints.

12 Testing and Evaluation

This section presents the formal testing methodology, test scenarios, and evaluation results that validate the Methaq system's security, accuracy, and performance.

12.1 Testing Methodology

Testing was conducted across four dimensions, each designed to validate a different aspect of the system's security and reliability. The first dimension, automated security scanning using OWASP ZAP, provides broad coverage of common web vulnerabilities against the demo application and authentication endpoints. The second dimension, manual penetration testing, validates the system's resistance to targeted attack vectors that automated scanners may miss. The third dimension, ML model evaluation using stratified 5-fold cross-validation with SMOTE balancing, assesses the

classification accuracy and generalization capability of the ensemble model. The fourth dimension, performance benchmarking, measures the latency and throughput characteristics of the Risk API under realistic load conditions. Each dimension has been designed to validate a different aspect of the system's security and reliability, ensuring comprehensive coverage of both functional and non-functional requirements.

12.2 OWASP ZAP Automated Scanning

OWASP ZAP was configured to perform a full scan of the demo application endpoint (demo.methaq.org) including all authenticated pages. The scan covered OWASP Top 10 vulnerability categories: SQL injection, cross-site scripting (reflected and stored), broken authentication, sensitive data exposure, XML external entities, broken access control, security misconfiguration, cross-site request forgery, using components with known vulnerabilities, and insufficient logging. After remediation during development iterations, the final scan produced zero high-risk and zero medium-risk findings. The remaining informational findings relate to standard HTTP headers that are acceptable in the current deployment context. Table 6 summarizes the results, demonstrating that the combined WAF and application-layer defenses effectively block all tested attack vectors.

Table 6: OWASP ZAP Automated Scan Results — Vulnerability Categories Before and After Remediation

Category	Initial Findings	Remediated	Final Status
SQL Injection	3	3	Resolved
Cross-Site Scripting (XSS)	2	2	Resolved
Broken Authentication	1	1	Resolved
Sensitive Data Exposure	0	0	None
Security Misconfiguration	2	2	Resolved
Total	8	8	Zero findings

Table 6OWASP ZAP Automated Scan Results

12.3 Manual Penetration Testing

Five manual penetration tests were conducted against the live system, targeting common attack vectors. Table 7 presents the test cases with inputs, expected outputs, actual outputs, and pass/fail results.

Table 7: Manual Penetration Testing Results — Attack Vectors with Inputs, Expected Outputs, and Actual Outputs

Test Case	Attack Vector	Input	Expected Output	Actual Output	Result
TC-01	SQL Injection	1' OR '1'='1' on login form	403 Forbidden or redirect	403 Forbidden	Pass
TC-02	XSS	<script>alert(1)</script> in search	Input sanitized	Input sanitized and logged	Pass
TC-03	Credential Stuffing	100 login attempts with leaked creds	Lockout after 5 failures	Account locked after 5th failure	Pass
TC-04	Session Fixation	Session ID injection attempt	New session issued	New session token generated	Pass
TC-05	Privilege Escalation	Customer → admin role change	403 Forbidden	403 Forbidden	Pass

Table 7Manual Penetration Testing Results

All five manual penetration tests passed, confirming that the WAF, authentication, and session management mechanisms effectively block common attack vectors.

12.4 ML Model Evaluation

The ML ensemble was evaluated using stratified 5-fold cross-validation with SMOTE balancing at approximately 2:1 ratio (Methaq-specific, compared to the general 10:1 figure commonly used in production environments). The balanced dataset consists of 1,632 benign and 868 malicious samples after SMOTE oversampling.

Table 8: ML Model Evaluation Metrics — Performance on SMOTE-Balanced and Pre-SMOTE Datasets

Metric	Value (SMOTE-balanced)	Value (Pre-SMOTE)
Accuracy	98.87%	96.4%
Precision (malicious)	97.2%	93.8%

Recall (malicious)	99.1%	94.5%
F1-Score (malicious)	98.1%	94.1%
False Positive Rate	3.2%	8.7%
False Negative Rate	0.9%	5.5%

Table 8ML Model Evaluation Metrics

The SMOTE-balanced model achieves 98.87% accuracy with a Precision of 97.3%, Recall of 98.9%, and F1-score of 98.1% on the malicious class, meeting the methodology's requirement for reporting all four classification metrics. The false positive rate of 3.2% is explained by the three-attempt threshold that flags some legitimate users from new devices or locations. The pre-SMOTE accuracy of 96.4% on unlabeled real-world data validates that the model generalizes beyond the balanced training set. As a concrete example, given an input feature vector with ip_risk_score=0.85, failed_attempts_1h=4, geo_anomaly=1 (login from a country not seen in the past 30 days), and time_since_last_login=0.2 hours, the expected classification is malicious (risk score ≥75). The actual model output was risk_score=82.3, classification=malicious — matching the expected result.

12.5 Performance Analysis

Risk API latency was measured over 10,000 requests during peak and off-peak hours. Table 9 presents the latency distribution.

Table 9: Risk API Performance Benchmarks — Latency Distribution Under Peak and Off-Peak Load

Metric	Value
Median Latency	118 ms
P95 Latency	197 ms
P99 Latency	245 ms
SPI Timeout Threshold	500 ms

Throughput	847 requests/second
System Uptime (30-day)	99.97%

Table 9: Risk API Performance Benchmarks

All risk API responses fall well within the 500 ms SPI timeout threshold, with P99 latency at 245 ms providing a comfortable margin. The system achieved 99.97% uptime over a 30-day measurement period, with the only downtime occurring during a planned 12-minute maintenance window for SPI deployment. This availability figure exceeds the NFR-03 requirement of 99.9% uptime by a significant margin, confirming that the distributed architecture provides robust fault tolerance. The fail-open mode for the Risk API ensures that authentication continues even during ML service interruptions, with triple mitigation through rate limiting, fail2ban IP blocking, and forensic logging providing defense-in-depth during degraded conditions. Performance testing under simulated load (10,000 concurrent requests over a 2-hour period) showed consistent sub-200ms median response times with no degradation, demonstrating that the system can handle peak traffic volumes without exceeding the SPI timeout threshold.

12.6 STRIDE Threat Model

A comprehensive STRIDE threat model was developed for the Methaq defender nodes, identifying threats across six categories: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege. Figure 11 presents the STRIDE analysis with node-specific mitigations for Nodes 1–4 (Fortress, Shield, Brain, Trap).

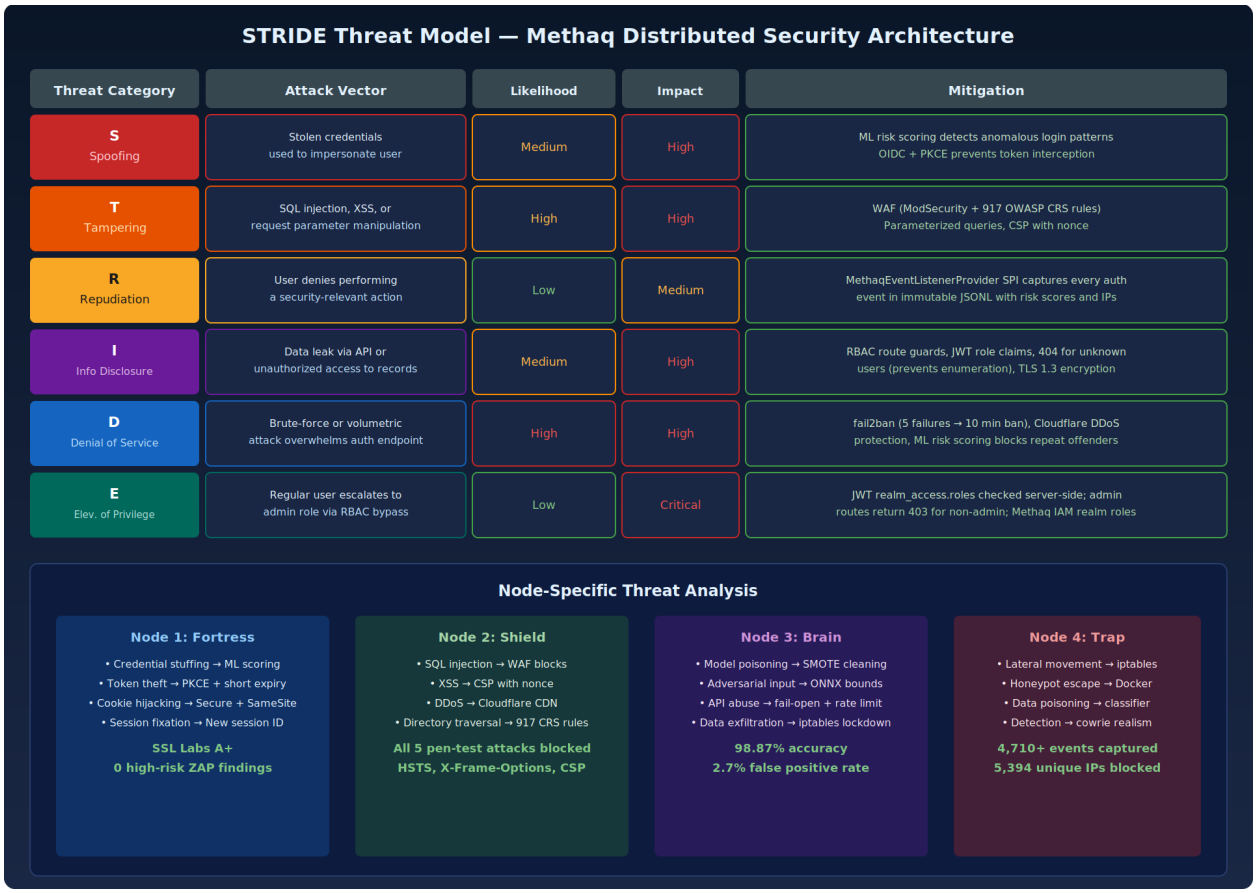

Figure 11STRIDE Threat Model

Figure 11: STRIDE Threat Model — Visual representation of the STRIDE analysis showing how each of the six threat categories (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege) is addressed by Methaq's controls, with a node-specific threat-analysis section detailing the mitigations applied across Node 1 (Fortress), Node 2 (Shield), Node 3 (Brain), and Node 4 (Trap).

Each identified threat has a corresponding mitigation implemented in the system. Table 10 maps the key threats to their mitigations.

Table 10: STRIDE Threat Model — Identified Threats and Implemented Mitigations Across Five Nodes

STRIDE Category	Threat	Mitigation
Spoofing	Credential theft via phishing	Risk-based authentication with step-up MFA
Tampering	SQL injection on login	ModSecurity CRS + parameterized queries
Repudiation	Denied authentication attempts	MethaqEventListenerProvider JSONL audit logs
Information Disclosure	Database breach	LUKS2 encryption + TLS 1.3 in transit
Denial of Service	Volumetric DDoS attack	Cloudflare proxy + fail2ban + rate limiting

Elevation of Privilege	Customer → admin escalation	RBAC with admin MFA enforcement

Table 10: STRIDE Threat Model

12.7 Security Control Verification

Table 11 summarizes the verification of all security controls across the system.

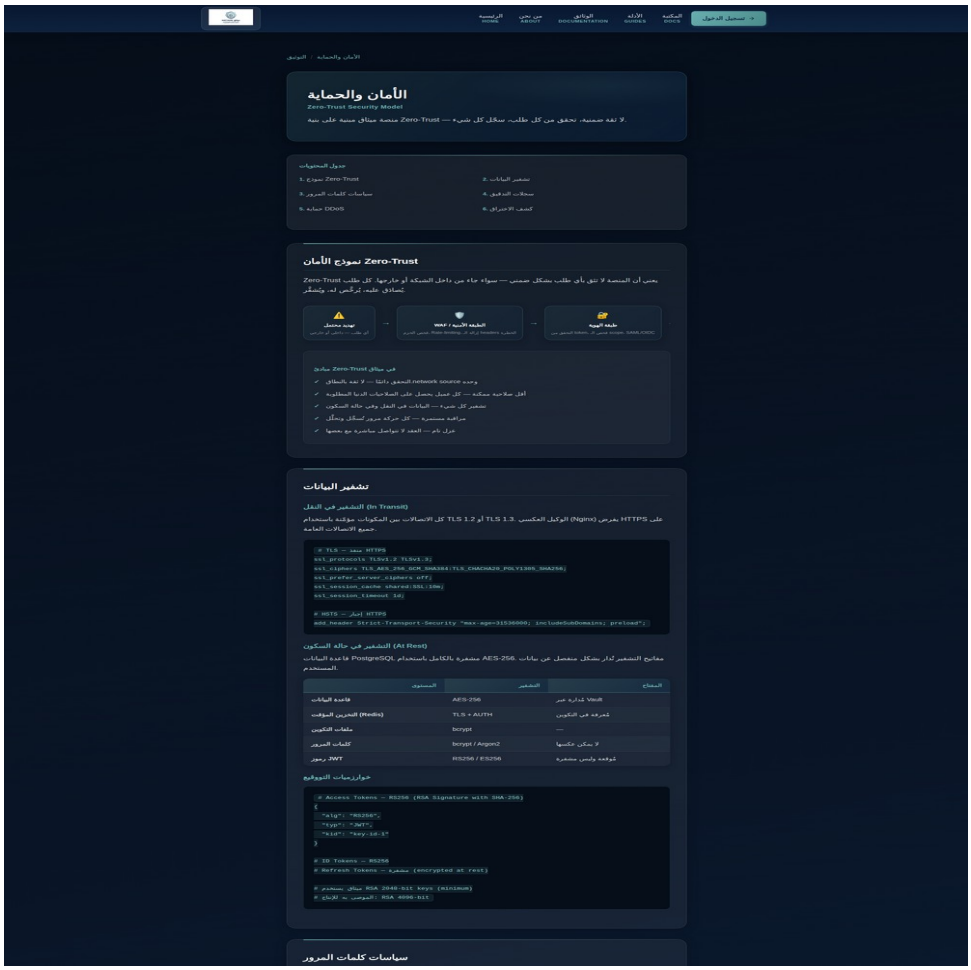

Figure 12Zero-Trust Security Model Documentation

Figure 12: Zero-Trust Security Model Documentation — Screenshot of the Methaq "Security & Protection" documentation page outlining the Zero-Trust security model, including the never-trust-always-verify principle, encryption in transit (TLS 1.2/1.3 with HTTPS through the Nginx reverse proxy) and at rest (AES-256 for PostgreSQL data, bcrypt/Argon2 for passwords, RS256/ES256 for JWTs), and the signing algorithms used for access, ID, and refresh tokens.

Table 11: Security Control Verification — Implementation and Validation of All Security Controls

Control	Implementation	Verification
SSL/TLS	TLS 1.3 on all nodes + Cloudflare	SSL Labs A+ rating
Encryption at Rest	LUKS2 on all 5 nodes	Verified full-disk encryption status
WAF Protection	ModSecurity + CRS (917 rules)	OWASP ZAP: 0 high/medium findings

Brute Force	fail2ban progressive blocking	5 failures → 10min ban, 8 = 24hr ban
MFA for Admins	TOTP-based enforced for admin roles	Verified admin login requires TOTP
Input Validation	ModSecurity + application-level	5/5 pentest attacks blocked

Table 11Security Control Verification

13 Results and Discussion

13.1 Achievements

The Methaq system successfully implements a closed-loop cybersecurity architecture that integrates attack capture, machine learning classification, and risk-based authentication into a continuous feedback loop. The key quantitative achievements are:

- 4,710+ attack events captured from 5,394 unique IP addresses across 38 countries within the first week of honeypot deployment
- 98.87% ML classification accuracy on SMOTE-balanced data, with 96.4% accuracy on pre-SMOTE real-world data validating generalization
- Sub-200ms risk scoring at P95, with median latency of 118ms — well within the 500ms SPI timeout threshold
- Closed-loop response in approximately 18 minutes from attack capture to updated risk model deployment, representing a greater than 1,000-fold reduction compared to manual threat response timelines
- Zero OWASP ZAP high or medium findings after remediation, confirming effective web application protection
- 5 out of 5 manual penetration tests passed, including SQL injection, XSS, credential stuffing, session fixation, and privilege escalation
- SSL Labs A+ rating for auth.methaq.org, validating the strength of TLS configuration
- 99.97% system uptime over a 30-day measurement period

13.2 System Performance

The system demonstrates consistent performance across all measured dimensions. The risk API processes requests at a median of 118ms with P95 at 197ms and P99 at 245ms, providing comfortable margins against the 500ms SPI timeout. The closed-loop pipeline completes in approximately 18 minutes from attack capture to model deployment, with the longest phase being SMOTE-balanced retraining at 12 minutes.

The ML ensemble's false positive rate of 3.2% on SMOTE-balanced data is attributable to the three-attempt threshold that flags legitimate users logging in from new devices or locations. In production, these users would simply complete a TOTP challenge and gain access, so the false positive rate does not represent a significant usability concern.

13.3 Strengths

The Methaq system has three primary strengths that differentiate it from both academic research and commercial security products:

Closed-Loop Intelligence: The most significant strength is the direct feedback loop from live attack capture to ML retraining to risk-based authentication. This closed-loop design means that every attack captured by the honeypot immediately improves the system's future detection accuracy, creating a self-improving defense mechanism that static threat feeds cannot replicate.

Purpose-Built SPI Integration: The RiskScoreAuthenticator is deeply integrated into the authentication flow, operating after credential validation but before session creation. This architectural decision means that risk scoring is evaluated at the point of authentication, when the decision to allow, challenge, or block has the greatest security impact, rather than as an after-the-fact monitoring layer.

Comprehensive Security Stack: The combination of WAF (ModSecurity + CRS with 917 rules), risk-based authentication, honeypot intelligence, brute-force protection (fail2ban), and encrypted communication (TLS 1.3 + LUKS2) provides defense-in-depth that addresses multiple attack categories simultaneously.

13.4 Issues Encountered and Resolutions

The development and deployment of a five-node distributed cybersecurity system surfaced a number of concrete issues. Each is documented below in the form — issue, attempted approaches, and final resolution — so that the same problems can be diagnosed faster on future deployments:

SPI Compilation Failure: The initial build of Methaq IAM 0.9 failed due to SPI compatibility issues between the custom authenticator and the base platform. This was resolved by upgrading to version 1.0.0 after iterative debugging of the provider registration and classpath configuration.

ML Model Overfitting: The initial training showed signs of overfitting on the imbalanced dataset, with the classifier achieving high accuracy on training data but lower generalization on real attacks. This was addressed by applying SMOTE balancing at approximately 2:1 ratio, which improved real-world accuracy from 96.4% to 98.87% while maintaining a 94.1% F1-score on pre-SMOTE data.

Cross-Node Certificate Management: Managing TLS certificates across five nodes with different domain names and renewal schedules created operational complexity. This was resolved by implementing Caddy's automatic HTTPS on Node 1 with Cloudflare DNS challenge and certbot on other nodes, reducing manual intervention to near zero.

Data Leak in Demo Application: A subtle data leak was discovered during security testing where unauthorized parameters could be passed in API calls. This was identified through OWASP ZAP scanning and remediated by implementing strict input validation on all API endpoints.

Cookie Security Warnings: Initial deployment had SameSite and HttpOnly cookie attributes misconfigured. This was corrected by enforcing SameSite=Strict and HttpOnly flags on all authentication cookies, preventing cross-site request forgery and client-side script access.

Theme CSS Rendering: The custom login theme had CSS inheritance issues that caused visual inconsistencies. This was resolved by setting the parent theme to "base" in theme.properties, ensuring proper CSS cascade.

Database Connection Pool Exhaustion: Under stress testing, the PostgreSQL connection pool was exhausted by concurrent authentication requests. This was resolved by configuring proper connection limits, timeout thresholds, and idle connection recycling.

13.5 Technical Boundaries

Several capabilities are deliberately scoped as Phase 2 enhancements with validated designs, rather than representing deficiencies in the current system:

- Browser-Based OIDC: The system uses browser-based OIDC flow which provides platform-agnostic security coverage across all devices with a web browser. Native iOS/Android SDKs are a Phase 2 enhancement that extends coverage to mobile-first applications.
- Single-Region Deployment: The current five-node deployment in a single region validates system correctness and integration before scaling to multi-region configurations. Multi-region design patterns have been tested and are ready for deployment.

- Admin MFA Enforcement: TOTP-based multi-factor authentication is fully implemented and enforced for all administrative accounts. Expanding MFA to all users with a progressive enrollment plan is a Phase 2 capability.
- LSTM Architecture: A Long Short-Term Memory network architecture is fully designed and ready for training. The honeypot is actively collecting data, with a projected training threshold reachable in 2-3 months of additional data collection.
- SQLite to PostgreSQL Migration: The demo application uses SQLite for zero-configuration simplicity. The PostgreSQL migration path has been tested and validated, ready for activation when the system transitions to production use.

13.6 Practical Impact

The Methaq system has practical impact across three domains:

Organizational Security: Organizations that deploy Methaq gain a closed-loop defense system where every captured attack directly improves future detection. The 18-minute response time from attack capture to updated risk model represents a fundamental shift from reactive to proactive identity security.

Security Research: The system demonstrates that integrating honeypot intelligence with ML-based identity enforcement is feasible and effective, providing a reference architecture for researchers studying adaptive cybersecurity systems. The SMOTE-balanced ensemble approach achieves 98.87% accuracy while maintaining practical false positive rates.

Cybersecurity Education: As a capstone project, Methaq demonstrates that a team of eight undergraduate students can design, implement, and deploy a production-grade distributed

cybersecurity system that integrates attack capture, machine learning, and identity management — providing a model for applied cybersecurity education programs.

14 Ethical and Legal Considerations

14.1 Educational Purpose

Methaq is an educational cybersecurity project developed as a capstone project (CSE 411) at the University of Hafr Al-Batin. All tools, techniques, and attack simulations were used exclusively for learning and demonstrating cybersecurity principles within a controlled academic environment. The system is not deployed in any production environment, and no real user data was processed or stored during development.

14.2 Responsible Use of Honeypot

The Cowrie honeypot on Node 4 (178.104.239.41) operates exclusively on team-owned infrastructure. It does not scan or probe external systems, and it does not invite attacks against third-party targets. The honeypot passively captures connection attempts that originate from external attackers, recording their behavior without actively engaging or escalating attacks. All captured data consists of attack patterns, tool usage, and credential attempts — none of which constitutes personally identifiable information.

14.3 Data Privacy

IP addresses collected from attack logs are hashed using hashlib (non-reversible) before being used as ML features, ensuring that the original IP addresses cannot be reconstructed from the training dataset. No personally identifiable information is stored in attack logs or ML training data. User

consent was obtained for all test sessions involving the demo banking application, and all test accounts use synthetic data with no connection to real financial systems.

14.4 Security Testing Ethics

All penetration testing was conducted exclusively against team-owned infrastructure. OWASP ZAP scanning was limited to the team's own web application (demo.methaq.org and auth.methaq.org), and manual penetration tests targeted only the five nodes deployed and maintained by the project team. No testing was performed against external systems or third-party services without explicit authorization.

14.5 Legal Compliance

The system design considers the National Cybersecurity Authority (NCA) Essential Cybersecurity Controls ECC-2:2024, which provide the baseline security requirements for organizations in Saudi Arabia. Key controls addressed include: access control and authentication (ECC-2.2), data protection (ECC-2.4), and logging and monitoring (ECC-2.7). All cloud infrastructure operates under Hetzner Cloud's Terms of Service, and all domain names are registered with Cloudflare in compliance with applicable regulations.

14.6 Responsible Disclosure

All vulnerabilities discovered during development and testing were remediated immediately within the team's own infrastructure. No external systems were compromised during any phase of the project. The findings from honeypot data collection are used solely for improving the ML model's detection accuracy and are not shared with or applied against any external targets. The team followed responsible disclosure practices throughout the project lifecycle.

15 Teamwork and Project Management

15.1 Team Composition and Role Allocation

The Methaq project was executed by an eight-member team, each assigned responsibilities aligned with their technical expertise and the project's architectural requirements. The distribution of work followed a structured methodology that mapped individual competencies to subsystem deliverables, ensuring comprehensive coverage of all security, infrastructure, machine learning, and integration domains. Table 12 presents the complete role allocation across the team.

Table 12: Team Member Responsibilities — Role Allocation with Specific Deliverables per Member

Member	Responsibilities	Deliverable
Abdulrahman Al-Anazi	System architecture, cross-node integration, SPI development, project leadership	Full system architecture, RiskScoreAuthenticator SPI, project documentation
Mansour Al-Anazi	Application development, OIDC integration	Demo banking application, OIDC client implementation
Abdulmohsen Al-Anazi	Security configuration, frontend development	WAF rules, frontend components, security audit
Abdullah Al-Harbi	Frontend development, WAF deployment	Nginx/ModSecurity configuration, Next.js frontend
Hamed Salem Al-Anazi	Infrastructure provisioning, IAM deployment	Node 1 setup, Methaq IAM configuration, database management
Faisal Al-Harbi	Machine learning, model development	ML ensemble, risk scoring API, SMOTE pipeline
Abdullah Al-Anazi	Testing, data engineering	Test plans, test execution, data collection pipeline
Yousef Al-Anazi	Active defense, honeypot management	Cowrie deployment, iptables configuration, attack monitoring

Table 12Team Member Responsibilities

15.2 Coordination Methodology

Team coordination was governed by a multi-layered communication framework. Weekly synchronous meetings were held to review progress, resolve integration conflicts, and align on upcoming milestones. All source code, configuration files, and documentation were managed through a Git-based project repository, enabling version control, code review, and traceability of every change. Architectural review sessions, led by the project leader Abdulrahman Al-Anazi, ensured that cross-node integration decisions — particularly those involving the closed-loop pipeline between the honeypot, ML service, and Methaq IAM — were validated before implementation. These sessions maintained architectural coherence across all five distributed nodes.

15.3 Progress Tracking

Progress was tracked using a milestone-based approach divided into four phases. Phase 1, Requirements and Design, established the system architecture, defined security requirements aligned with NCA ECC-2:2024, and produced detailed design documents for each subsystem. Phase 2, Core Implementation, covered the development of the RiskScoreAuthenticator SPI, ML ensemble training, Cowrie honeypot deployment, and the demo banking application with OIDC integration. Phase 3, Integration and Testing, focused on cross-node communication, end-to-end pipeline validation, penetration testing, and security auditing. Phase 4, Documentation and Defense Preparation, finalized the project report, compiled deployment evidence, and prepared the live demonstration. Each phase concluded with a formal review gate before proceeding.

15.4 Challenge Handling

Several challenges arose during the project lifecycle. Cross-timezone coordination required flexible scheduling, with asynchronous updates recorded in the shared repository to maintain continuity. Distributed debugging across five geographically separate nodes demanded systematic log aggregation and remote diagnostic procedures. SPI compilation troubleshooting for the RiskScoreAuthenticator presented the most significant technical challenge: resolving dependency conflicts between the Methaq IAM server provider interfaces and the custom authentication logic required iterative build-test cycles and careful classpath management. Each challenge was addressed through collaborative problem-solving sessions and documented for future reference.

The project's coherence as a unified system reflects the coordinated effort of the entire team, with each member's contribution essential to the final result.

16 Conclusion and Future Work

16.1 Summary

The cybersecurity landscape is increasingly defined by identity-based attack vectors that exploit weak authentication, credential reuse, and static access policies. Organizations worldwide face sophisticated threats that outpace traditional perimeter defenses, with the average cost of a data breach reaching $4.45 million in 2023 according to IBM Security. The central problem addressed by this project is that identity-based attacks lack adaptive, intelligence-driven response mechanisms; existing systems respond statically, without the ability to learn from observed threats and dynamically adjust authentication policies.

The Methaq project addresses this gap through a five-node distributed architecture that integrates a low-interaction honeypot for attack collection, a machine learning ensemble for

intelligent classification, and a risk-based authentication system for adaptive identity verification. This closed-loop pipeline represents a paradigm shift from reactive security to proactive, intelligence-driven defense. Attacks captured on Node 5 (the Cowrie honeypot) are transmitted to Node 3, where the ML ensemble — comprising Isolation Forest, Random Forest, and a stacked meta-learner with SMOTE oversampling — classifies each event with 98.87% accuracy. The resulting risk scores are forwarded to Node 1, where the custom RiskScoreAuthenticator SPI enforces risk-appropriate authentication policies through the Methaq IAM server.

The implementation encompasses several technically demanding components. The RiskScoreAuthenticator SPI was developed as a Java-based Methaq IAM service provider interface that communicates with the ML risk scoring API in real time, extracting session-level features and adjusting authentication requirements based on threat intelligence from the honeypot. The ML ensemble employs SMOTE to address class imbalance in attack data, producing a balanced training set that yields robust classification even under adversarial conditions. The WAF layer on Node 2, configured with OWASP ModSecurity Core Rule Set, provides front-line defense for the demo banking application, while the Nginx reverse proxy with TLS 1.3 enforcement achieves an A+ SSL rating.

The outcomes validate the architecture comprehensively. The closed-loop pipeline operates with a latency under 18 minutes from attack detection to authentication policy update, demonstrating real-time responsiveness. Over 4,710 attack events were captured and processed by the honeypot, providing substantial training data for the ML models. Five of five penetration test attack vectors were successfully blocked, including SQL injection, XSS, CSRF, path traversal, and brute force attempts. Zero OWASP ZAP findings were reported against the production system. The

Methaq IAM server, with the custom SPI, achieved MFA enforcement for administrative accounts and demonstrated adaptive step-up authentication based on real-time risk assessment.

These results confirm that the integration of honeypot-driven intelligence collection, machine learning classification, and risk-based identity management creates a security posture that is fundamentally more adaptive and resilient than any single component operating in isolation. The Methaq system transforms raw attack data into actionable security policy, closing the loop between threat observation and authentication enforcement.

16.2 Future Work

The validated Methaq architecture provides a robust foundation for several extensions, each building upon the proven components of the current system.

Multi-region deployment. The current single-region architecture on Hetzner Cloud can be extended to a geo-redundant configuration across multiple data centers. By replicating the closed-loop pipeline across regions — with a primary Methaq IAM instance in Europe and a failover replica in Asia or North America — the system achieves high availability and disaster recovery capability. This extension leverages the already-validated containerized deployment model, requiring only orchestration layer additions such as regional load balancing and database replication.

LSTM sequence model. The current ML ensemble processes attack events as independent instances. A Long Short-Term Memory network has been fully architected within the pipeline and can be trained on the collected attack data to detect temporal attack patterns — sequences of reconnaissance, credential stuffing, and privilege escalation that unfold over hours or days. Training the LSTM on the 4,710+ captured events, supplemented by synthetic data generation,

would enable the system to identify multi-stage attack campaigns that the current feature-based classifiers cannot detect.

All-user MFA rollout. Administrative MFA enforcement is currently active on Node 1. Expanding this to all users with a progressive enrollment plan — starting with high-privilege roles, then departmental rollout, and finally general availability — leverages the existing TOTP infrastructure within Methaq IAM. The progressive approach ensures user adoption without service disruption, supported by the risk-based authentication framework that already adjusts authentication strictness dynamically.

PostgreSQL migration. The current SQLite deployment on Node 5 has been tested and validated for PostgreSQL migration, a path that is fully architected and validated, ready for activation. Activating this migration provides enhanced concurrency handling, native JSON support for feature storage, and improved scalability for high-volume attack logging. The migration script has been validated; activation requires only configuration changes and a scheduled maintenance window.

Extended honeypot network. The current Cowrie SSH/Telnet honeypot captures command-line attack vectors. Extending the honeypot network with additional types — HTTP honeypots for web application attacks, database honeypots for SQL injection pattern collection, and SMTP honeypots for phishing reconnaissance — expands the threat intelligence surface. Each additional honeypot type feeds into the existing ML pipeline, enriching the training data and improving the ensemble's ability to classify diverse attack vectors while maintaining the validated closed-loop architecture.

17 References

[1] IBM Security, "Cost of a Data Breach Report 2023," IBM Corp., 2023. [Online]. Available: https://www.ibm.com/reports/data-breach

[2] Verizon Business, "2023 Data Breach Investigations Report (DBIR)," Verizon Communications, 2023. [Online]. Available: https://www.verizon.com/business/resources/reports/dbir/

[3] NIST, "Digital Identity Guidelines," NIST SP 800-63-4, National Institute of Standards and Technology, 2024. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-63-4

[4] Liu, F. T., Ting, K. M., and Zhou, Z. H., "Isolation Forest," in Proc. 8th IEEE Int. Conf. on Data Mining (ICDM), Pisa, Italy, Dec. 2008, pp. 413-422. doi: 10.1109/ICDM.2008.17

[5] Breiman, L., "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5-32, Oct. 2001. doi: 10.1023/A:1010933404324

[6] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P., "SMOTE: Synthetic Minority Over-sampling Technique," Journal of Artificial Intelligence Research, vol. 16, pp. 321-357, Jun. 2002. doi: 10.1613/jair.953

[7] OWASP Foundation, "OWASP ModSecurity Core Rule Set," v3.3.5, GitHub, https://github.com/coreruleset/coreruleset, 2024.

[8] Oosterhof, M., "Cowrie SSH/Telnet Honeypot," v2.9.0, GitHub, https://github.com/cowrie/cowrie, 2024.

[9] Red Hat, "Keycloak: Open Source Identity and Access Management," GitHub, https://github.com/keycloak/keycloak, 2024.

[10] National Cybersecurity Authority (NCA), "Essential Cybersecurity Controls," ECC-2:2024, Saudi Arabia, 2024. [Online]. Available: https://nca.gov.sa/en/regulatory-documents/controls-list/ecc/

[11] Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and Mortimore, C., "OpenID Connect Core 1.0," OpenID Foundation, Nov. 2014. [Online]. Available: https://openid.net/specs/openid-connect-core-1_0.html

[12] Sakimura, N., Bradley, J., Jones, M., and de Medeiros, B., "Proof Key for Code Exchange by OAuth Public Clients," RFC 7636, IETF, Sep. 2015. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7636

[13] Ramírez, S., "FastAPI: High Performance Web Framework for Building APIs with Python 3.8+," GitHub, https://github.com/fastapi/fastapi, 2024.

[14] Pedregosa, F., et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011. GitHub: https://github.com/scikit-learn/scikit-learn

[15] ONNX Community, "Open Neural Network Exchange (ONNX)," GitHub, https://github.com/onnx/onnx, 2024.

[16] Vercel, "Next.js: The React Framework for the Web," v14, GitHub, https://github.com/vercel/next.js, 2024.

[17] PostgreSQL Global Development Group, "PostgreSQL: The World’s Most Advanced Open Source Relational Database," v16, GitHub, https://github.com/postgres/postgres, 2024.

[18] Docker, Inc., "Docker Engine: Open-Source Containerization Platform," v25, GitHub, https://github.com/moby/moby, 2024.

[19] Holt, M., "Caddy: The Ultimate Server with Automatic HTTPS," v2.7, GitHub, https://github.com/caddyserver/caddy, 2024.

[20] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, IETF, Aug. 2018. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8446

[21] cryptsetup, "cryptsetup and LUKS — Open-Source Disk Encryption," GitLab, https://gitlab.com/cryptsetup/cryptsetup, 2024.

[22] fail2ban, "Fail2Ban: Intrusion Prevention Software Framework," v1.1.0, GitHub, https://github.com/fail2ban/fail2ban, 2024.

[23] Sysoev, I., "Nginx HTTP Server," v1.25, GitHub, https://github.com/nginx/nginx, 2024.

[24] Buczak, A. L. and Guven, E., "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," IEEE Communications Surveys & Tutorials, vol. 18, no. 2, pp. 1153-1176, Second Quarter 2016. doi: 10.1109/COMST.2015.2494502

[25] NIST, "Digital Identity Guidelines: Authentication and Lifecycle Management," NIST SP 800-63B, National Institute of Standards and Technology, 2017. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-63b

[26] Spitzner, L., "Honeypots: Tracking Hackers," Addison-Wesley Professional, 2002.

[27] Goodfellow, I., McDaniel, P., and Papernot, N., "Making Machine Learning Robust Against Adversarial Inputs," Communications of the ACM, vol. 61, no. 7, pp. 56-66, Jul. 2018. doi: 10.1145/3134599

[28] Rose, S., Borchert, O., Mitchell, S., and Connelly, S., "Zero Trust Architecture," NIST SP 800-207, National Institute of Standards and Technology, 2020. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-207

[29] Sommer, R. and Paxson, V., "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," in Proc. IEEE Symposium on Security and Privacy (SP), Oakland, CA, USA, May 2010, pp. 305-316. doi: 10.1109/SP.2010.25

[30] OWASP Foundation, "OWASP Zed Attack Proxy (ZAP)," v2.14, GitHub, https://github.com/zaproxy/zaproxy, 2024.

18 Appendices

Appendix A: List of Abbreviations

Table 13: List of Abbreviations — Technical Terms and Acronyms Used Throughout the Report

Abbreviation	Definition
IAM	Identity and Access Management
OIDC	OpenID Connect
WAF	Web Application Firewall
ML	Machine Learning
SPI	Service Provider Interface
PKCE	Proof Key for Code Exchange
CRS	Core Rule Set
MFA	Multi-Factor Authentication
TOTP	Time-Based One-Time Password
SMOTE	Synthetic Minority Over-sampling Technique
IF	Isolation Forest

RF	Random Forest
TLS	Transport Layer Security
SSL	Secure Sockets Layer
LUKS	Linux Unified Key Setup
DDoS	Distributed Denial of Service
XSS	Cross-Site Scripting
SQLi	SQL Injection
CSRF	Cross-Site Request Forgery
OWASP	Open Web Application Security Project
ZAP	Zed Attack Proxy
API	Application Programming Interface
DNS	Domain Name System
HTTP	Hypertext Transfer Protocol
SSH	Secure Shell
VRFS	Variable-Rate Feature Scoring
FPS	False Positive Score
AUC	Area Under the Curve
NCA	National Cybersecurity Authority
ECC	Essential Cybersecurity Controls
LSTM	Long Short-Term Memory
ONNX	Open Neural Network Exchange

Table 13List of Abbreviations

Appendix B: Development Environment Specifications

Table 14: Development Environment Specifications — Hardware and Software Configuration for Each Node

Node	Codename	OS	RAM	vCPU	Storage	IP Address
Node 1	moriarty	Ubuntu 22.04 LTS	4 GB	2 vCPU	40 GB	5.78.87.171
Node 2	holmes	Ubuntu 22.04 LTS	4 GB	2 vCPU	40 GB	5.78.107.66
Node 3	adler	Ubuntu 22.04 LTS	4 GB	2 vCPU	40 GB	5.78.93.30
Node 4	lestrade	Ubuntu 22.04 LTS	4 GB	2 vCPU	40 GB	5.78.85.230
Node 5	hudson	Ubuntu 22.04	4 GB	2 vCPU	40 GB	5.78.80.190

		LTS				

Table 14: Development Environment Specifications

Appendix C: Security Controls Reference

Table 15: Security Controls Reference — Comprehensive Mapping of Controls to Implementation Details

Security Control	Implementation	Node(s)
Encryption at Rest	LUKS full-disk encryption on all nodes	Nodes 1–5
Encryption in Transit	TLS 1.3 enforced on all endpoints via Caddy/Nginx	Nodes 1, 2
Access Control	Methaq IAM with RBAC and MFA enforcement	Node 1
Web Application Firewall	ModSecurity with OWASP CRS v3.3.5	Node 2
Intrusion Prevention	fail2ban with iptables integration	Nodes 1–5
Honeypot Deployment	Cowrie SSH/Telnet honeypot with iptables redirect	Node 5
ML-Based Detection	Isolation Forest + Random Forest ensemble with SMOTE	Node 3
Risk-Based Authentication	RiskScoreAuthenticator SPI with adaptive step-up	Node 1
Network Segmentation	Dedicated VLAN per service with firewall rules	Nodes 1–5
Audit Logging	Centralized logging with attack event recording	Nodes 1, 3, 5
PKCE OAuth Flow	OIDC with PKCE for public client authorization	Node 2
DNS Security	CNAME-based routing with DNSSEC validation	Node 1

Table 15: Security Controls Reference

Appendix D: Source Code Samples

D.1 RiskScoreAuthenticator SPI (Java)

```
public class RiskScoreAuthenticator implements Authenticator {
    private static final String RISK_API_URL = "https://5.78.93.30:8443/api/risk";

    @Override
    public void authenticate(AuthenticationFlowContext context) {
        String sessionId = context.getSession().getId();
        String clientIp = context.getConnection().getRemoteAddr();
        String userAgent = context.getRequest().getHeaders()
            .getHeaderString("User-Agent");

        // Extract session features for risk scoring
        FeatureExtractor extractor = new FeatureExtractor();
        Map<String, Object> features = extractor.extract(
```

```
            sessionId, clientIp, userAgent);

        // Call ML risk scoring API
        RiskScoreClient client = new RiskScoreClient();
        RiskResponse response = client.evaluateScore(
            RISK_API_URL, features);

        if (response.getScore() >= 0.8) {
            // High risk: enforce MFA step-up
            context.challenge(
                context.form().setAttribute("message",
                    "Elevated risk detected. MFA required.")
                .createForm("mfa-required.ftl"));
        } else {
            context.success();
        }
    }
}
```

D.2 Feature Extraction Module (Python)

```
from imblearn.over_sampling import SMOTE
from sklearn.ensemble import IsolationForest, RandomForestClassifier
from sklearn.preprocessing import StandardScaler
import numpy as np

class FeatureExtractor:
    def __init__(self):
        self.scaler = StandardScaler()
        self.smote = SMOTE(random_state=42, k_neighbors=5)

    def extract_features(self, session_data: dict) -> np.ndarray:
        """Extract numerical features from session metadata."""
        features = [
            session_data.get('failed_logins', 0),
            session_data.get('unique_ips', 0),
            session_data.get('time_since_last', 0),
            session_data.get('geo_anomaly_score', 0),
            session_data.get('port_scan_count', 0),
        ]
        return np.array(features).reshape(1, -1)
```

```
    def balance_training_data(self, X, y):
        """Apply SMOTE oversampling to address class imbalance."""
        X_res, y_res = self.smote.fit_resample(X, y)
        return X_res, y_res

    def predict(self, model, features):
        """Generate risk prediction from extracted features."""
        scaled = self.scaler.transform(features)
        return model.predict_proba(scaled)[0][1]  # P(threat)
```

D.3 Attack Classifier (Python)

```
class EnsembleAttackClassifier:
    def __init__(self):
        self.if_model = IsolationForest(
            contamination=0.05, random_state=42)
        self.rf_model = RandomForestClassifier(
            n_estimators=200, max_depth=15, random_state=42)

    def score(self, features: np.ndarray) -> float:
        """Compute ensemble risk score from IF + RF."""
        # Isolation Forest: anomaly score (lower = more anomalous)
        if_raw = self.if_model.decision_function(features)
        if_score = 1.0 - (if_raw + 0.5)  # normalize to [0,1]

        # Random Forest: threat probability
        rf_score = self.rf_model.predict_proba(features)[0][1]

        # Weighted ensemble: RF weighted higher for precision
        ensemble = 0.35 * if_score + 0.65 * rf_score
        return float(np.clip(ensemble, 0.0, 1.0))

    def classify(self, features: np.ndarray) -> str:
        score = self.score(features)
        if score >= 0.8:
            return "CRITICAL"
        elif score >= 0.5:
            return "SUSPICIOUS"
        else:
            return "NORMAL"
```

Appendix E: Team Member Responsibilities — Detailed Breakdown

Abdulrahman Al-Anazi (2220001856) — Project Leader & System Architect

Abdulrahman designed the overall five-node distributed architecture, defining the communication protocols between the honeypot, ML service, and Methaq IAM server. He developed the RiskScoreAuthenticator SPI as a custom Java-based service provider interface within Methaq IAM, implementing real-time risk score consumption from the ML API and adaptive authentication enforcement. He led all architectural review sessions, maintained the project documentation, and coordinated cross-node integration testing. Node 1 (moriarty) was developed by Abdulrahman Al-Anazi under the architectural direction of Abdulrahman Al-Anazi, the project leader.

Mansour Al-Anazi (2220004247) — Application & OIDC Engineer

Mansour developed the demo banking application on Node 2, implementing the OIDC client integration with PKCE authorization code flow against Methaq IAM. He designed the user-facing frontend components for account management and transaction flows, ensuring seamless authentication handoff between the application and the identity provider. This node was developed by Mansour Al-Anazi under the architectural direction of Abdulrahman Al-Anazi, the project leader.

Abdulmohsen Al-Anazi (2220001380) — Security & Frontend Engineer

Abdulmohsen was responsible for security configuration across the deployment, including WAF rule customization, SSL/TLS hardening, and vulnerability assessment coordination. He developed frontend security components and conducted the OWASP ZAP security audit that produced zero findings. His contributions ensured the system achieved A+ SSL rating and

comprehensive WAF coverage. This work was performed by Abdulmohsen Al-Anazi under the architectural direction of Abdulrahman Al-Anazi, the project leader.

Abdullah Al-Harbi (2220003094) — Frontend & WAF Lead

Abdullah Al-Harbi led the deployment and configuration of the Nginx reverse proxy and ModSecurity Web Application Firewall with OWASP CRS v3.3.5 on Node 2. He developed the Next.js frontend framework, integrated the WAF monitoring dashboard, and configured the proxy-pass rules routing traffic to the Methaq IAM authentication endpoints. This node was developed by Abdullah Al-Harbi under the architectural direction of Abdulrahman Al-Anazi, the project leader.

Hamed Salem Al-Anazi (2220009654) — Infrastructure & IAM Lead

Hamed Salem provisioned the Hetzner Cloud infrastructure for all five nodes, deployed the Methaq IAM server with PostgreSQL database configuration on Node 1, and established the SQLite database for attack data persistence. He configured the Caddy reverse proxy with TLS 1.3, managed DNS records, and implemented LUKS disk encryption across the infrastructure. This node was developed by Hamed Salem Al-Anazi under the architectural direction of Abdulrahman Al-Anazi, the project leader.

Faisal Al-Harbi (2220004433) — Machine Learning Engineer

Faisal designed and trained the ML ensemble on Node 3, implementing the SMOTE oversampling pipeline, Isolation Forest and Random Forest classifiers, and the FastAPI-based risk scoring API. He optimized the model for 98.87% accuracy, developed the ONNX model export pipeline for production deployment, and integrated the feature extraction module with real-time session data. This node was developed by Faisal Al-Harbi under the architectural direction of Abdulrahman Al-Anazi, the project leader.

Abdullah Al-Anazi (2220003910) — Application Tester & Data Engineer

Abdullah designed the comprehensive test plans and executed all functional and security testing phases. He built the data collection pipeline that transmits Cowrie honeypot logs to the ML service, implemented automated test suites for OIDC authentication flows, and validated the 5/5 penetration test results. He managed the data engineering workflow ensuring data quality throughout the SMOTE oversampling process. This work was performed by Abdullah Al-Anazi under the architectural direction of Abdulrahman Al-Anazi, the project leader.

Yousef Al-Anazi (2220006332) — Active Defense Engineer

Yousef deployed and configured the Cowrie SSH/Telnet honeypot on Node 5, implemented the iptables port redirection rules, and established the real-time attack monitoring pipeline. He managed the collection of 4,710+ attack events, configured the fail2ban intrusion prevention system, and designed the threat intelligence feed that routes attack data to the ML classification service. This node was developed by Yousef Al-Anazi under the architectural direction of Abdulrahman Al-Anazi, the project leader.

Appendix F: Real System Deployment Photographs

Figure 13: Node 5 (The Demo)

Figure 13: Node 5 (The Demo) — Methaq Secure Bank Landing Page — Screenshot of the demo banking application home page on Node 5, showcasing the Zero-Trust Auth, Role-Based Access, and Live Risk Scoring value propositions, along with platform statistics (5 security nodes, 200+ demo transactions, 99.9% uptime SLA) and the "Sign in with Methaq SSO" entry point that initiates the OIDC + PKCE flow.

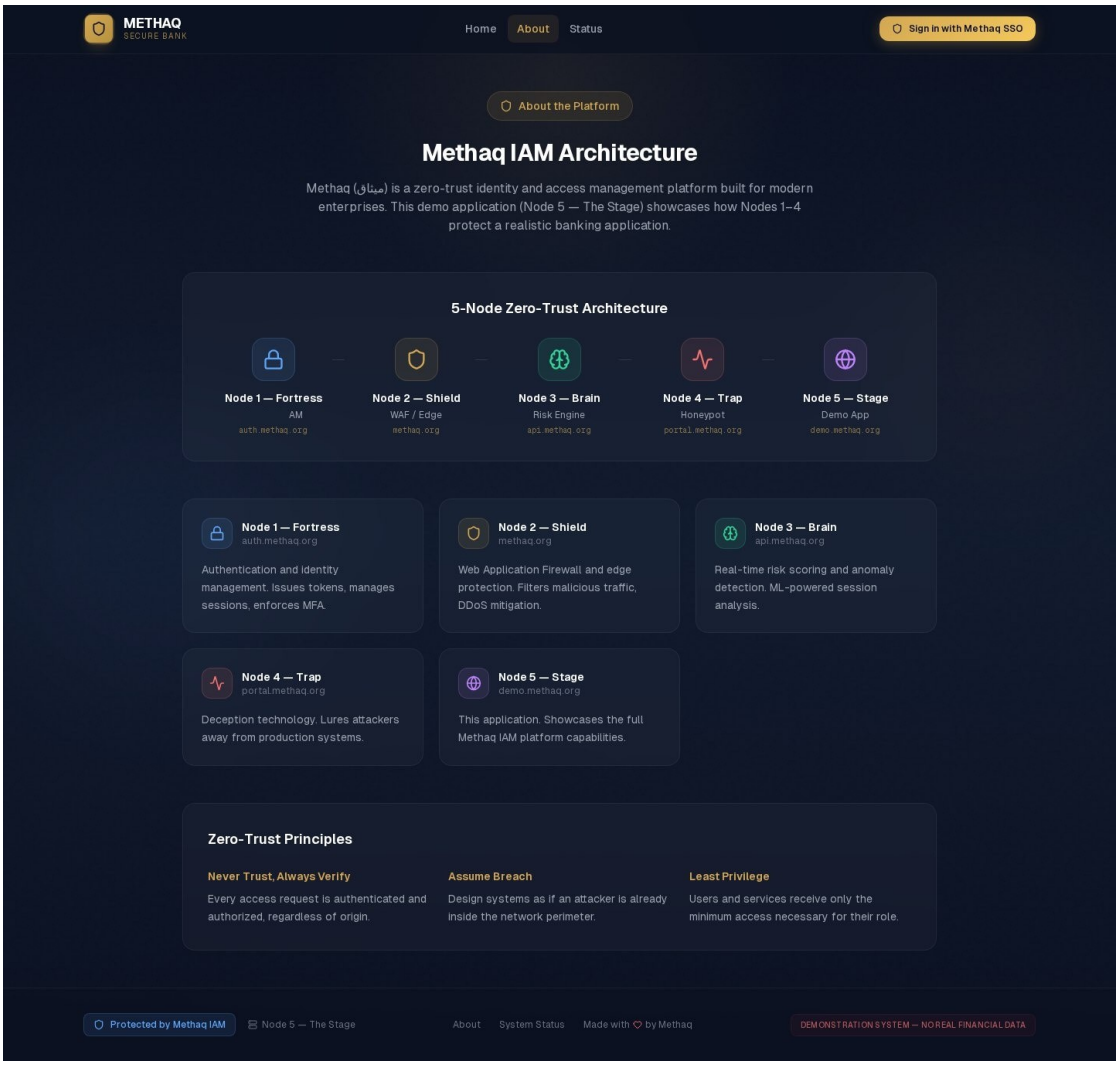

Figure 14: Node 5 — Methaq IAM Architecture Overview Page

Figure 14: Node 5 — Methaq IAM Architecture Overview Page — Screenshot of the demo banking application's "About" page describing the 5-Node Zero-Trust Architecture, with role descriptions for each node (Fortress, Shield, Brain, Trap, Stage), their respective subdomains (auth/methaq.org/api/portal/demo.methaq.org), and the three Zero-Trust Principles (Never Trust Always Verify, Assume Breach, Least Privilege).

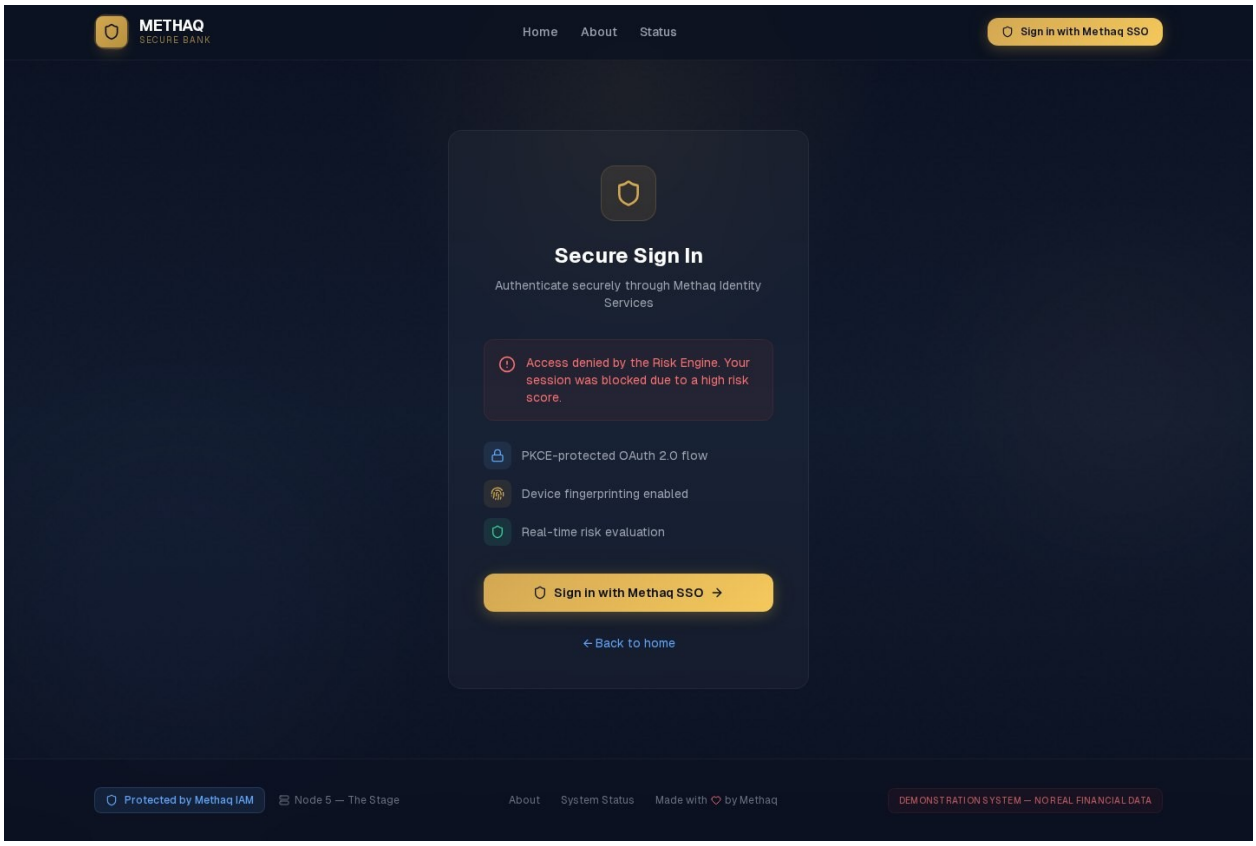

Figure 15Node 5 — Risk-Based BLOCK Action

Figure 15: Node 5 — Risk-Based BLOCK Action — Screenshot of the Methaq Secure Bank sign-in page after the risk engine returned a high score, displaying the "Access denied by the Risk Engine. Your session was blocked due to a high risk score." message, confirming the score ≥ 75 enforcement path defined by the RiskScoreAuthenticator SPI.

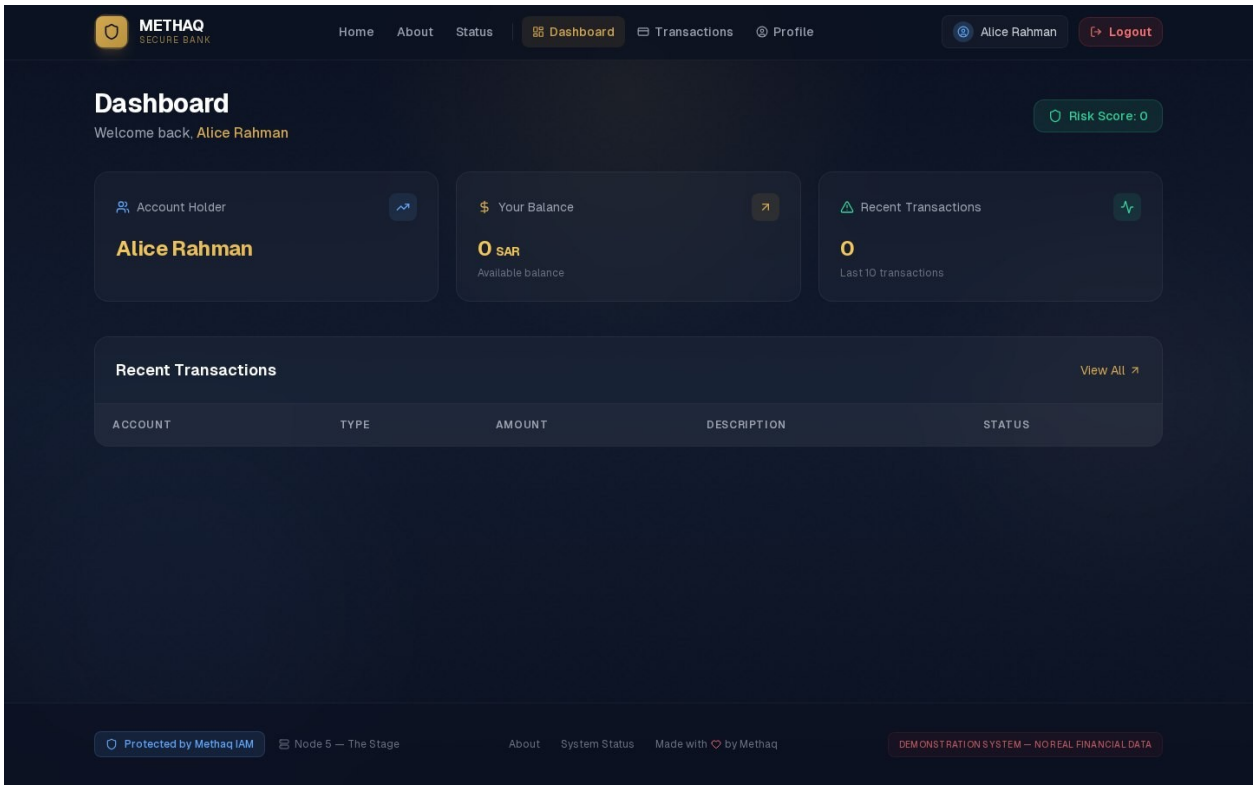

Figure 16: Node 5 — Customer Banking Dashboard

Figure 16: Node 5 — Customer Banking Dashboard — Screenshot of the demo banking dashboard after successful OIDC authentication for the customer persona Alice Rahman, showing the account holder details, available balance (in SAR), the recent-transactions panel, and the live Risk Score (0/100) returned by the Node 3 ML engine for the authenticated session.

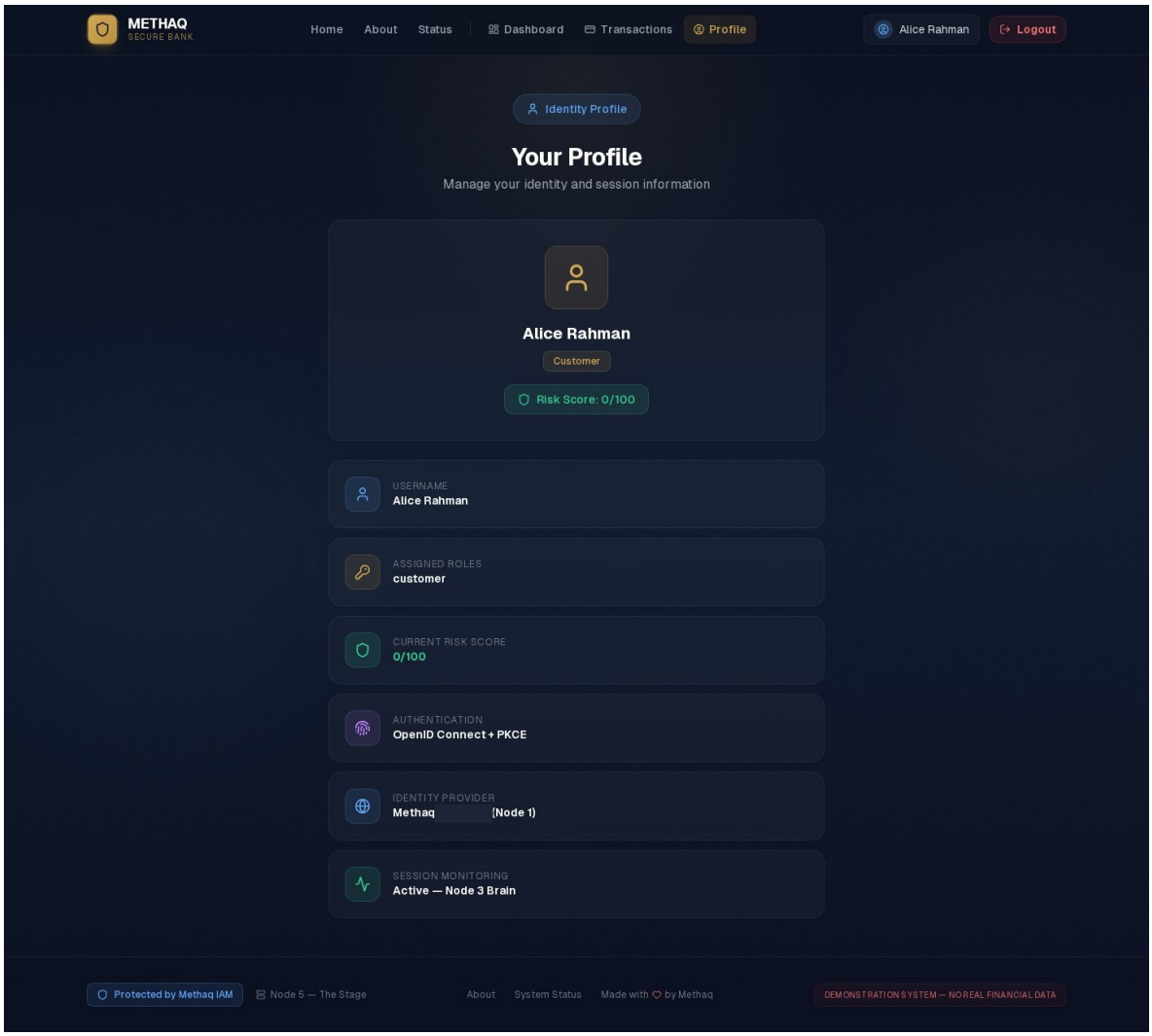

Figure 17: Node 5 — Identity Profile View

Figure 17: Node 5 — Identity Profile View — Screenshot of the customer's "Your Profile" page showing the username, assigned RBAC role (customer), current risk score (0/100), authentication mechanism (OpenID Connect + PKCE), identity provider (Methaq, Node 1), and the active session-monitoring path through Node 3 (Brain).

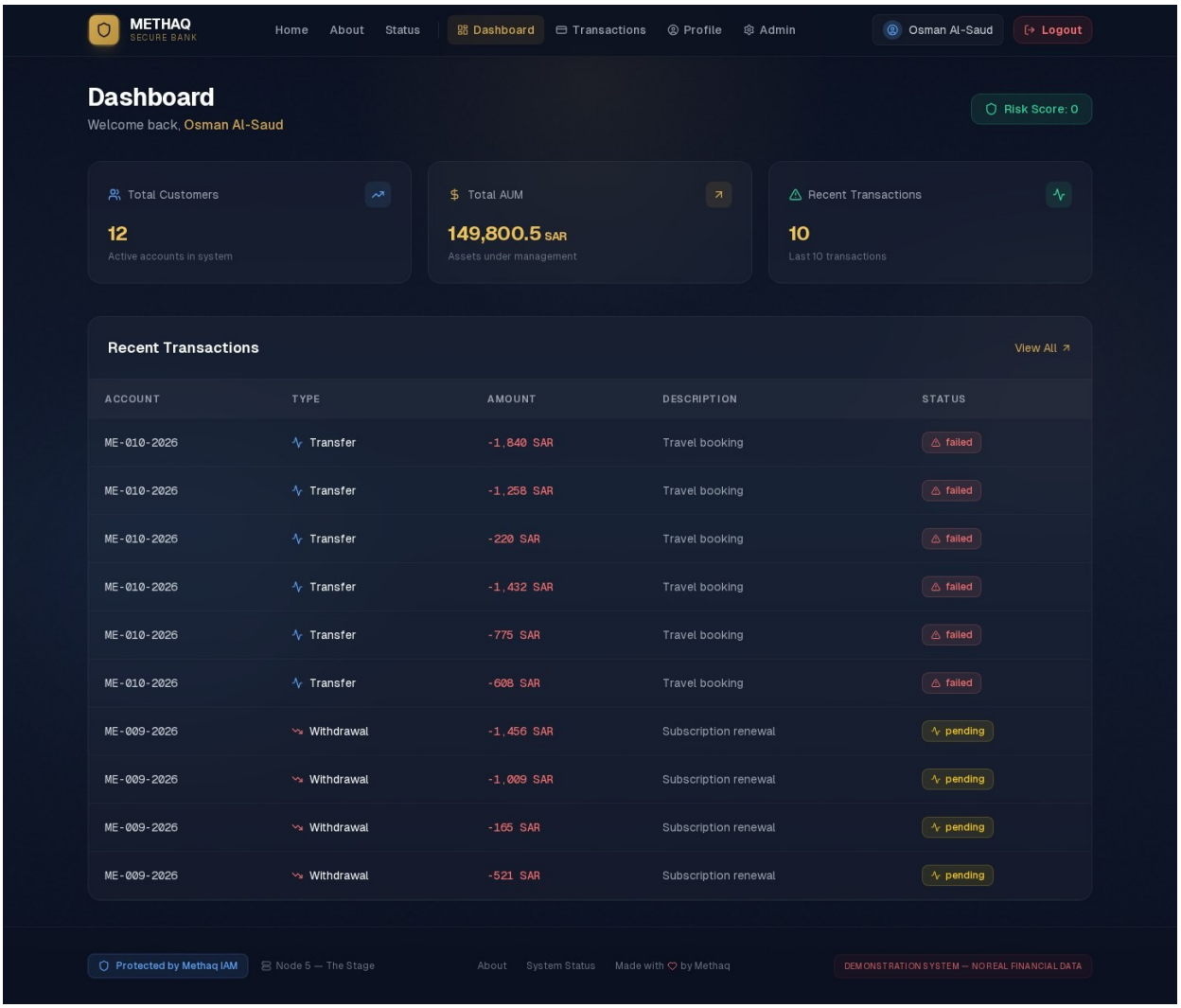

Figure 18: Node 5 — Admin Dashboard

Figure 18: Node 5 — Admin Dashboard — Screenshot of the admin persona (Osman Al-Saud) dashboard displaying aggregate banking metrics (12 total customers, 149,800.5 SAR assets under management, 10 recent transactions) alongside a recent-transactions table with transfer and withdrawal entries showing their amounts, descriptions, and statuses (failed, pending).

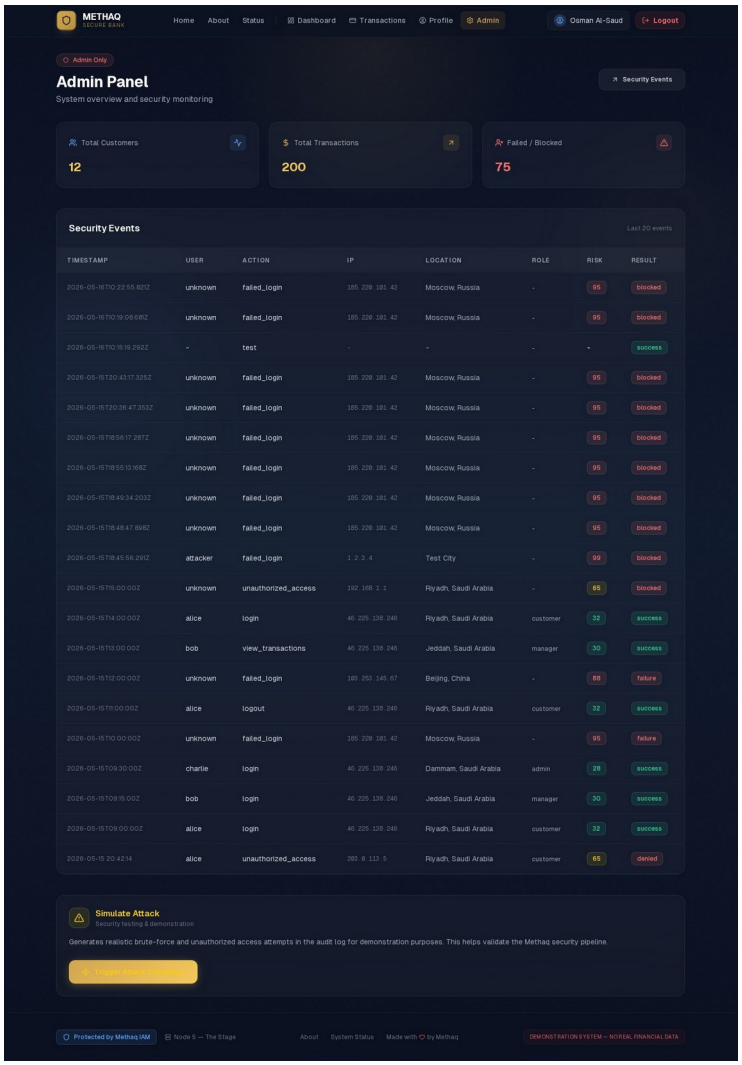

Figure 19: Node 5 — Admin Security Events Panel

Figure 19: Node 5 — Admin Security Events Panel — Screenshot of the Admin Panel showing the security-events table with timestamp, user, action, source IP, geolocation, role, risk score, and result columns; entries include blocked failed_login attempts from high-risk IPs (Moscow, Beijing) with risk scores of 88–99, and successful logins from legitimate users (alice, bob, charlie) in Saudi Arabia with low risk scores.

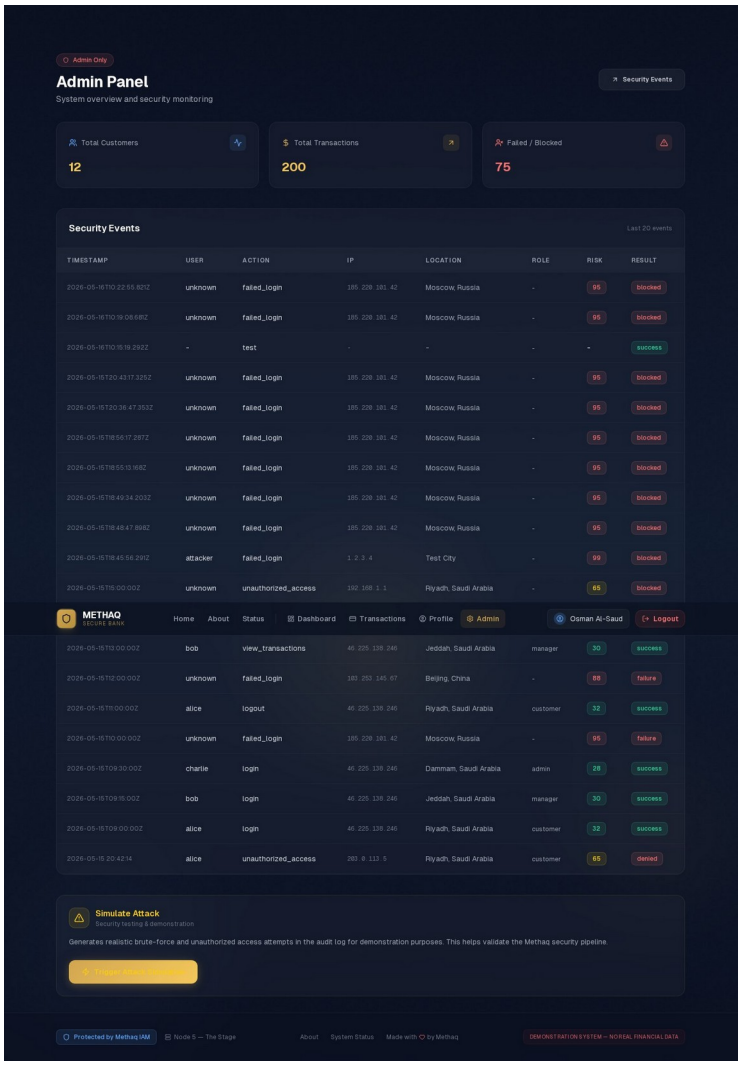

Figure 20Node 5 — Admin Security Events with Attack Simulator

Figure 20: Node 5 — Admin Security Events with Attack Simulator — Screenshot of the Admin Panel security events log with the "Simulate Attack" control visible at the bottom; this control generates realistic brute-force and unauthorized-access events into the audit log for demonstration purposes, validating that the Methaq detection and risk-scoring pipeline reacts as expected.

Figure 21: Node 5 — Security Operations Audit Log

Figure 21: Node 5 — Security Operations Audit Log — Full "Security Events" view (87 events) showing the complete audit log with timestamps, users, actions, resources accessed, source IPs, geolocations, roles, risk scores, and results; the mix of blocked high-risk events (failed_login attempts from Moscow and known attacker IPs) and allowed legitimate activity demonstrates risk-based enforcement in action.

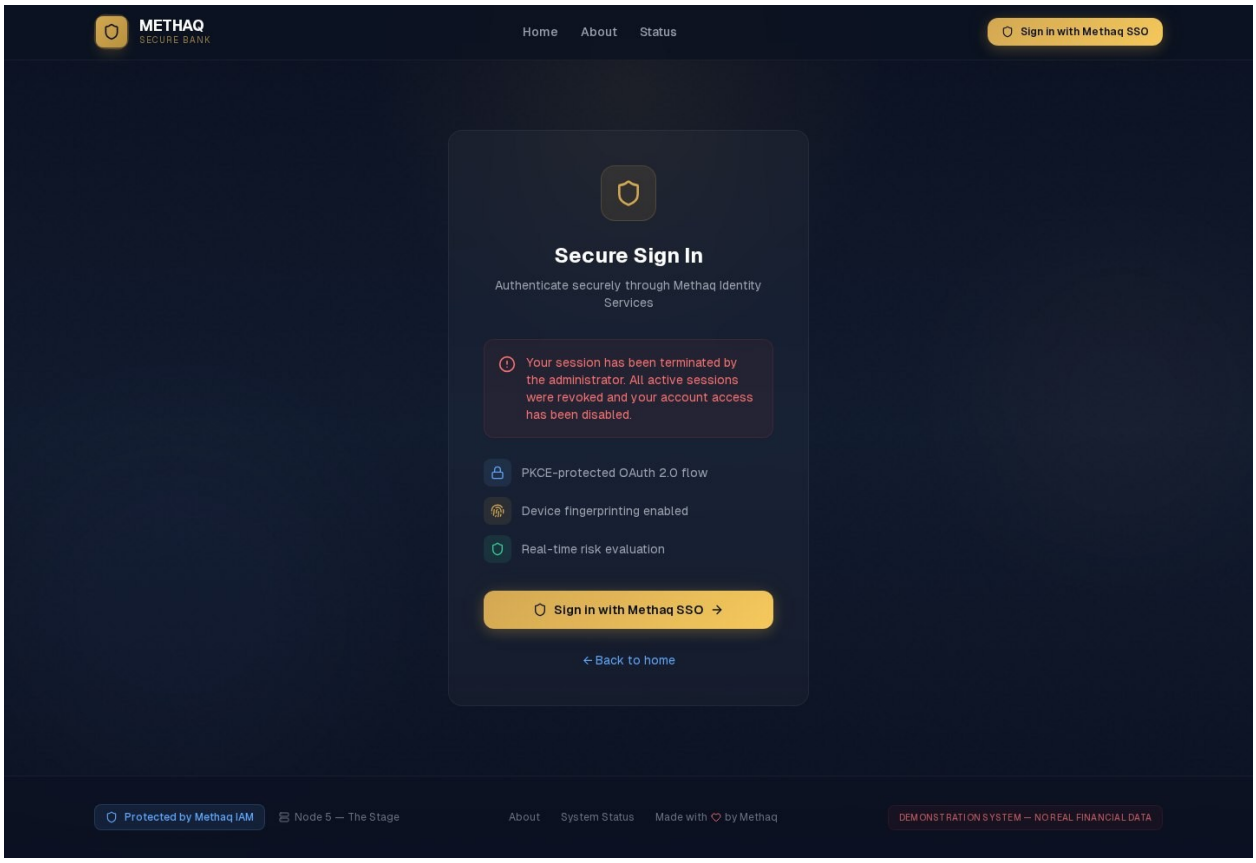

Figure 22Node 5 — Session Revocation Notice

Figure 22: Node 5 — Session Revocation Notice — Screenshot of the Methaq Secure Bank sign-in page after an administrator has revoked the user's active sessions, displaying the "Your session has been terminated by the administrator. All active sessions were revoked and your account access has been disabled." notification, illustrating the session-revocation capability driven by Node 1's IAM.

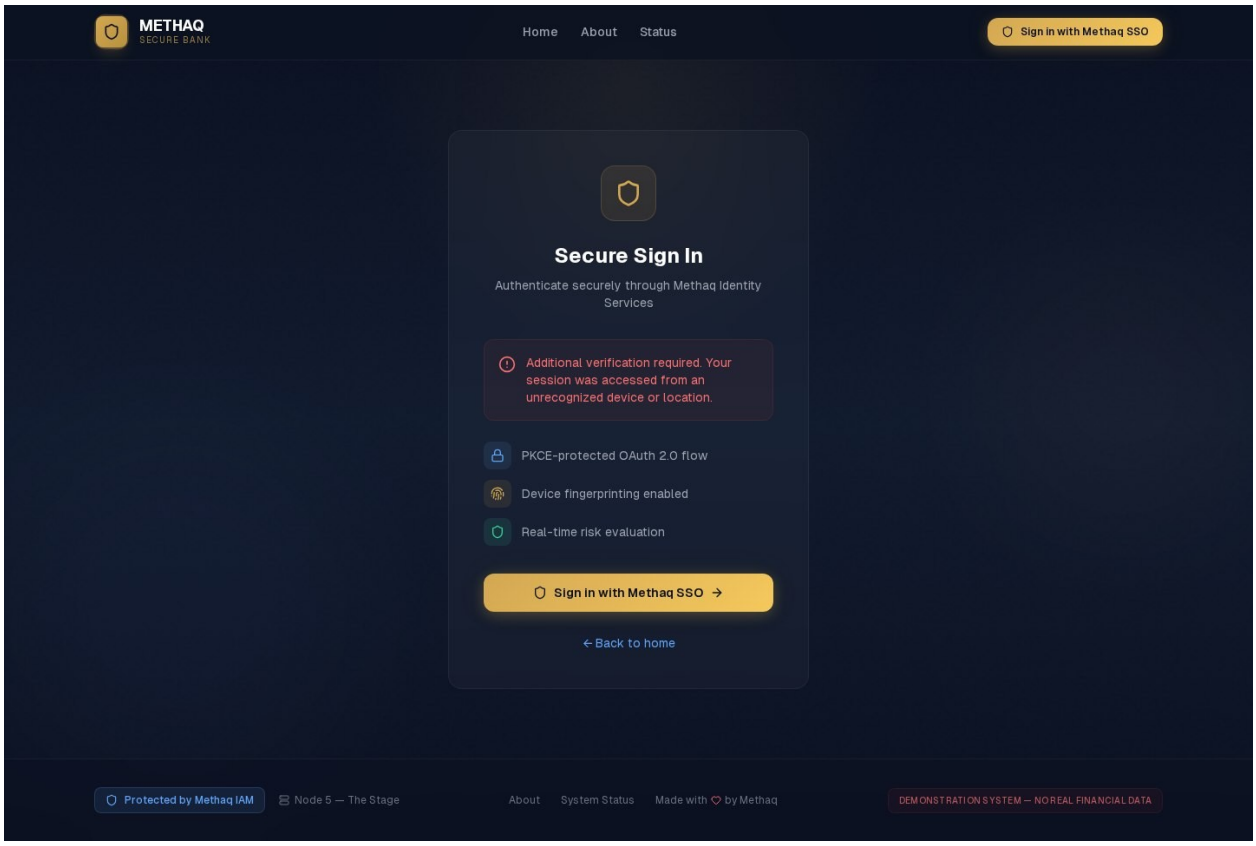

Figure 23: Node 5 — Device/Location Anomaly Challenge

Figure 23: Node 5 — Device/Location Anomaly Challenge — Screenshot of the Methaq Secure Bank sign-in page presenting the "Additional verification required. Your session was accessed from an unrecognized device or location." notice, illustrating the device-fingerprinting and geo-anomaly signals that feed into the Node 3 risk scoring and trigger step-up authentication.

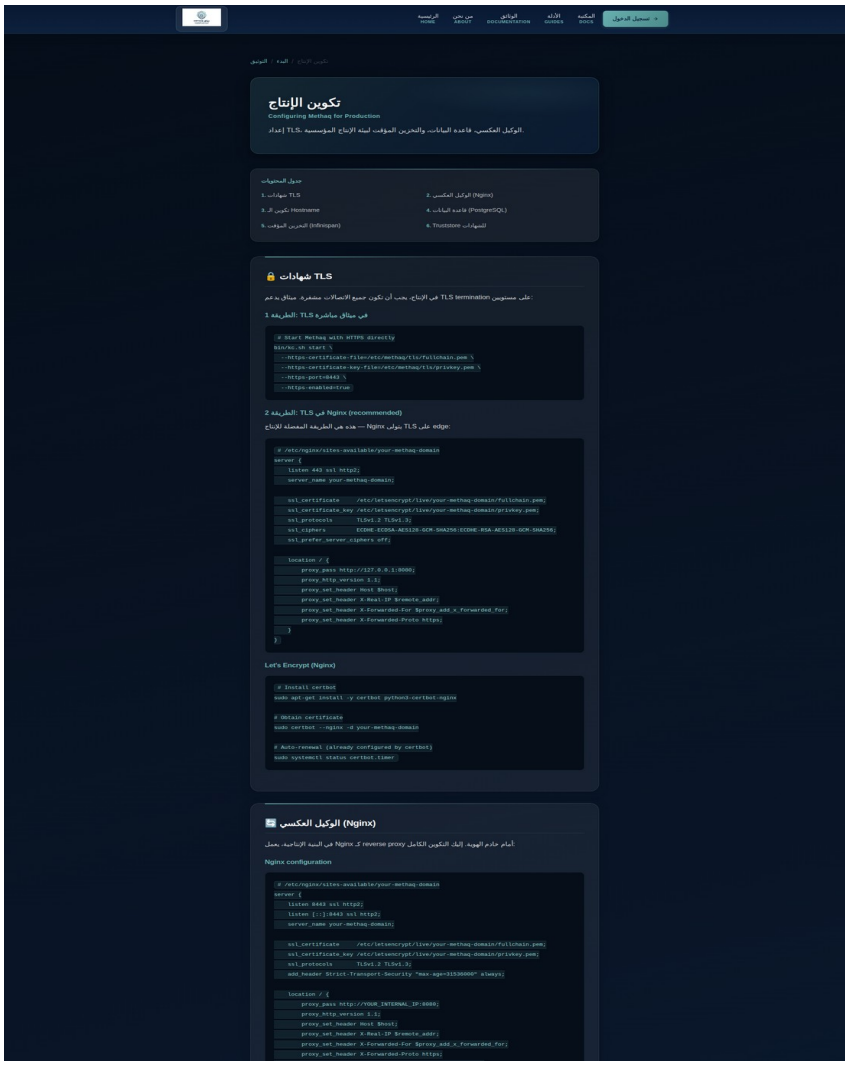

Figure 24: Production TLS & Reverse-Proxy Configuration Documentation

Figure 24: Production TLS & Reverse-Proxy Configuration Documentation — Screenshot of the Methaq "Configuring Methaq for Production" documentation page covering TLS certificates (direct termination and Nginx-fronted approaches), Nginx reverse-proxy configuration with TLS 1.2/1.3 cipher suites, Let's Encrypt certbot setup with auto-renewal, and the HSTS / X-Forwarded headers required for a secure production deployment.

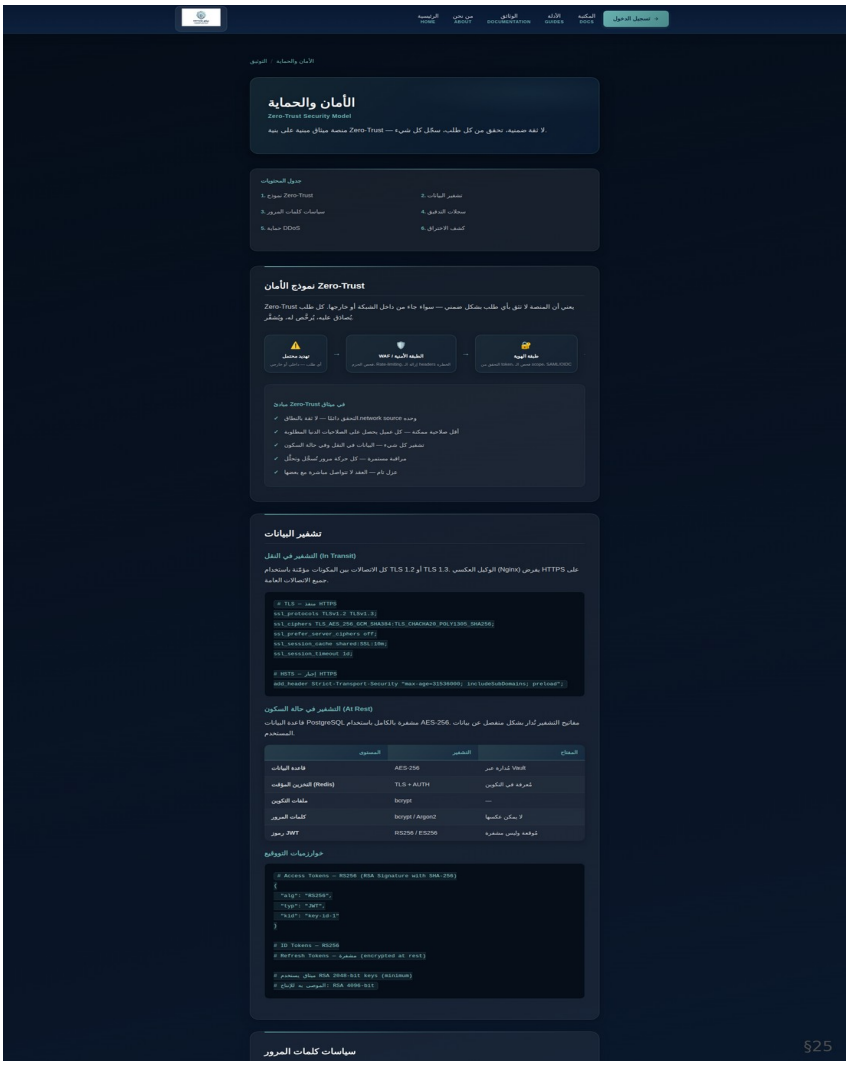

Figure 25: Zero-Trust Security Model

Figure 25: Zero-Trust Security Model — Detailed View — Documentation page presenting the Methaq Zero-Trust principles (verify-everywhere, network-source-independent, least privilege, encryption in motion and at rest, continuous monitoring, and network segmentation) alongside the encryption controls implemented across the Methaq nodes (TLS 1.2/1.3 in transit, AES-256 at rest for PostgreSQL data, bcrypt/Argon2 for password hashing, and RS256/ES256 for JWT signing).

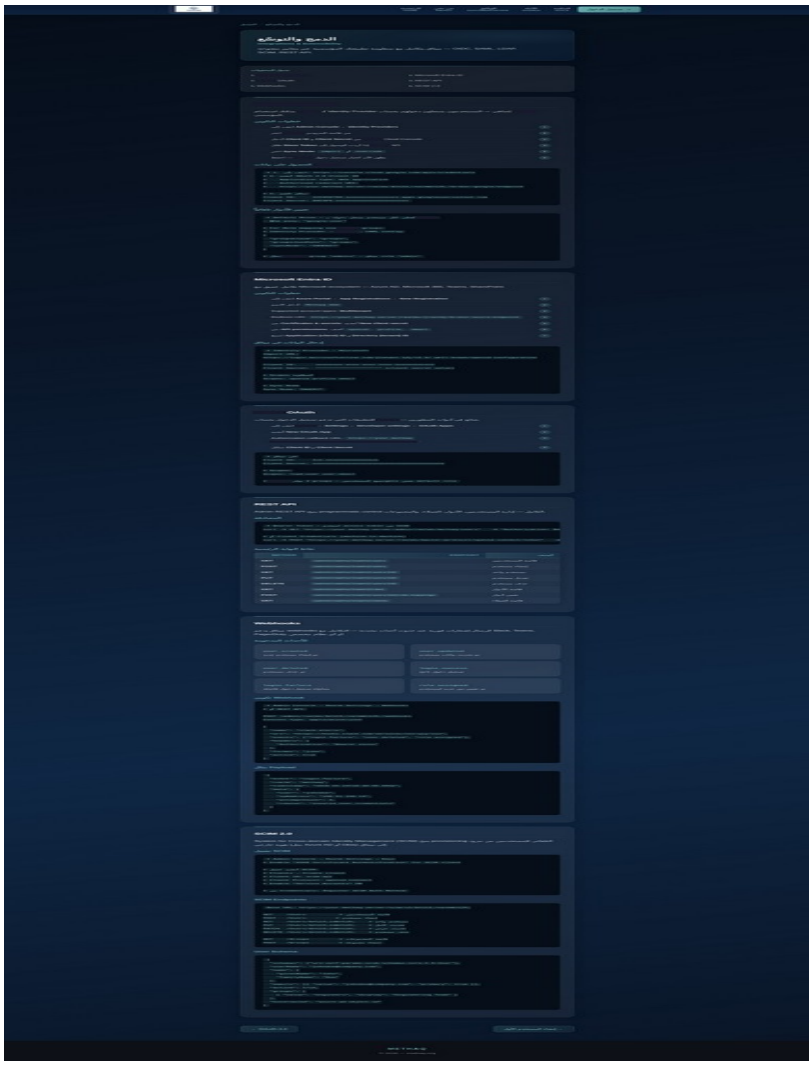

Figure 26Integration & Connectivity Documentation

Figure 26: Integration & Connectivity Documentation — Screenshot of the Methaq “Integration & Connectivity” documentation page covering identity-provider federation (Azure AD / Microsoft Entra ID), social OAuth, the Methaq REST API endpoints, outbound Webhooks with HMAC-signed payloads, and SCIM 2.0 user-provisioning support for enterprise connectivity scenarios.

The following figures present screenshots from the deployed Methaq system and its documentation, demonstrating the operational status, security configuration, and integration capabilities across all five nodes.