BRITISH CONGRESS

INTERDISCIPLINARY SCIENTIFIC RESEARCH & PRACTICES -VI

June 18-21, 2026



London, UK

PROCEEDINGS BOOK

Editor
Dawid WORTLEY

ISBN - 979-8-89695-480-4

A PROPOSAL FOR A SECURE AND OBSERVABLE MICROSERVICES ARCHITECTURE FOR AI-POWERED E-COMMERCE APPLICATIONS

Dr. Onur Polat[1], Ms. Nurhat Baydağ[2], Dr. Esra Söğüt[3]

[1]Department of Computer Engineering, Bingöl University, Bingöl, Türkiye;

0000-0001-9313-4910

[1]Department of Computer Engineering, Bingöl University, Bingöl, Türkiye;

0009-0006-0471-8708

[3]Department of Computer Engineering, Faculty of Technology, Gazi University, Ankara, Türkiye

0000-0002-0051-2271

ABSTRACT

This study proposes a microservices-based architecture that addresses the requirements of security, scalability, and observability in AI-powered e-commerce applications. In this study, an e-commerce system decoupled from the traditional monolithic structure is designed as a set of components comprising a web application, an AI service layer, an SQL Server database, a Redis cache and state management system, a Prometheus metrics collection infrastructure, and a Grafana monitoring dashboard. At the core of the proposed architecture lies a service approach based on the Model Context Protocol (MCP), which carries context between large language models (LLMs) and external data sources. Through this layer, sentiment analysis, question-answering, and content processing functions are executed in an isolated manner. The system is further supported by prompt injection prevention, data masking, DDoS/DoS mitigation, rate limiting, vulnerability scanner detection, bot blocking, an XSS sanitization process, and custom security metrics. The resulting architecture offers an integrated solution that applies a defense-in-depth approach against the new attack surfaces created by AI services. The study presents a system design based on engineering practice, and more comprehensive experimental validation with quantitative performance results is planned for the future.

Keywords: Microservices architecture, AI security, MCP, prompt injection, e-commerce security, observability

ÖZET

Bu çalışma, yapay zeka destekli e-ticaret uygulamalarında güvenlik, ölçeklenebilirlik ve gözlemlenebilirlik gereksinimlerini birlikte ele alan mikroservis tabanlı bir mimari önermektedir. Çalışmada, geleneksel monolitik yapıdan ayrıştırılmış bir e-ticaret sistemi; web uygulaması, yapay zeka servis katmanı, SQL Server veri tabanı, Redis önbellek ve durum yönetimi, Prometheus metrik toplama altyapısı ve Grafana izleme panelinden oluşan bileşenler halinde kurgulanmıştır. Önerilen mimarinin merkezinde, büyük dil modelleri ile dış veri kaynakları arasında bağlam taşıyan Model Bağlam Protokolü (Model Context Protocol - MCP) tabanlı bir servis yaklaşımı yer almaktadır. Bu katman üzerinden yorum analizi, soru-cevap ve içerik işleme işlevleri izole biçimde yürütülmektedir. Sistem ayrıca komut enjeksiyonu engelleme, veri maskeleme, DDoS/DoS etkisini azaltma, hız sınırlama, zafiyet tarayıcı tespiti, bot engelleme, XSS temizlik süreci ve özel güvenlik metrikleri ile desteklenmiştir. Elde edilen mimari, yapay zeka servislerinin doğurduğu yeni saldırı yüzeylerine karşı savunma derinliği yaklaşımını uygulayan bütünleşik bir çözüm sunmaktadır. Çalışma mühendislik uygulamasına dayalı bir sistem tasarımı sunmakta olup, gelecekte nicel performans sonuçları ile daha kapsamlı deneysel doğrulama yapılması planlanmaktadır.

Anahtar Kelimeler: Mikroservis mimarisi, yapay zeka güvenliği, MCP, komut enjeksiyonu, e-ticaret güvenliği, gözlemlenebilirlik

1. Introduction

The widespread adoption of large language models (LLMs) in areas such as e-commerce, customer experience, and content analysis has significantly altered the architectural and security requirements of traditional web applications. In particular, functions such as sentiment analysis, summarization of customer feedback, and natural language-based interactions provide high added value to applications while also creating new attack surfaces.

Embedding Artificial Intelligence (AI) functions directly into the main application in monolithic architectures can lead to various issues regarding maintenance, scaling, fault isolation, and the enforcement of security policies [1]. Therefore, handling AI services within a separate microservice layer offers a more suitable approach in terms of both operational flexibility and security control.

The aim of this study is to propose a secure microservice architecture for AI-enabled e-commerce platforms and to systematically describe the security components used in this architecture. In this context, the study integrates application layer security, data privacy, and observability mechanisms into a single architectural framework using an AI service layer based on the Model Context Protocol (MCP) approach. The main contributions of this study can be summarized under three headings:

• An MCP-based microservices architecture that separates AI functions from the main e-commerce application has been implemented.

• Multi-layered security mechanisms, such as prompt injection, DLP, DDoS mitigation, bot detection, browser detection, and content sanitization, have been integrated.

• A monitoring approach has been introduced that enhances operational visibility of security incidents through Prometheus and Grafana-based observability.

2. System Architecture

The proposed system consists of six core components designed according to the principle of separation of concerns (Figure 1). These components are the web application, the AI microservice, the SQL Server database, the Redis layer, Prometheus, and Grafana. These components are run separately within a Docker-based container architecture, thereby enhancing portability and manageability.

The web application manages user interactions and core e-commerce operations. AI-related functions are not executed directly within this layer; instead, they are processed through an independent service called McpService. This allows model changes, service updates, or AI-related error conditions to be isolated from the main application.

While SQL Server ensures the persistent storage of relational data, Redis is used both as a distributed cache and for rate limiting, blacklisting, and maintaining information on malicious activity [2,3]. Prometheus collects system metrics, while Grafana transforms these metrics into operational monitoring dashboards [4,5].

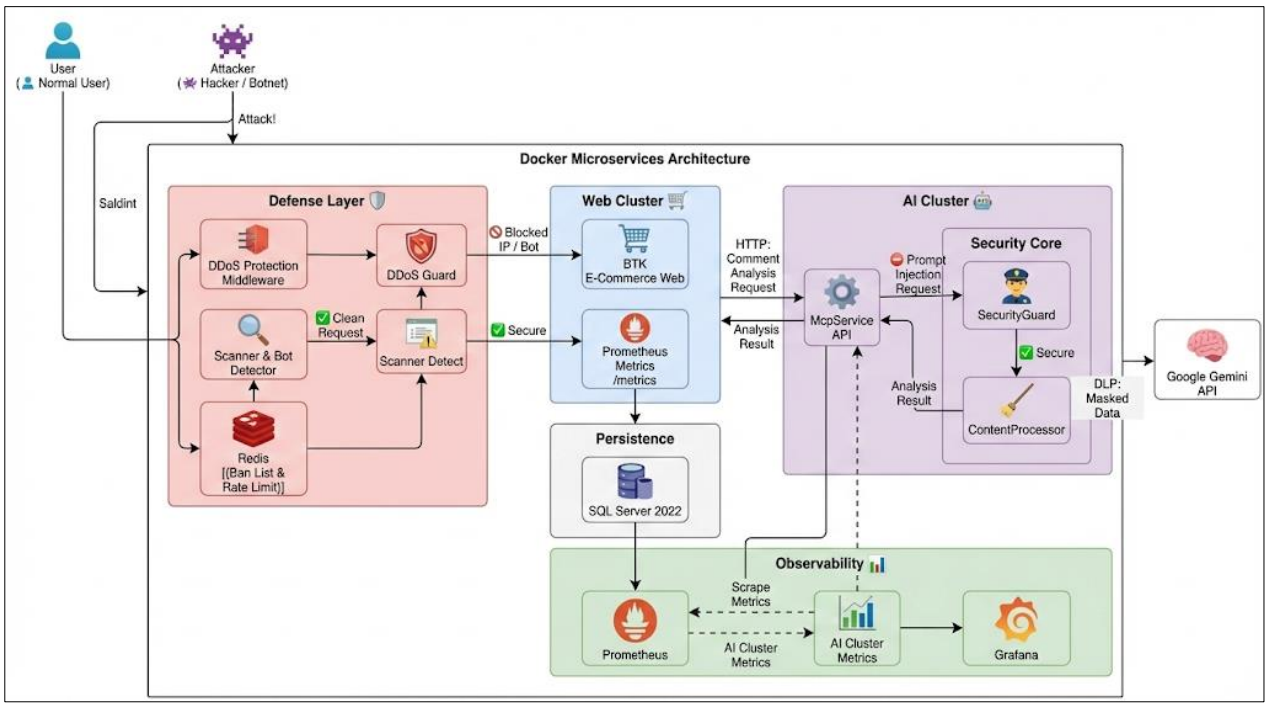


Figure 1. Diagram of the system architecture

3. MCP-Based AI Service Layer

In this study, the MCP approach was implemented by running the LLM (e.g., Google Gemini) through a controlled context provider service, rather than connecting it directly to the database or application logic. In this approach, the McpService retrieves the necessary content from the database, preprocesses it, and feeds the model only with the context it requires.

One advantage of this design is that it provides isolation, thereby establishing a clear boundary between the web application and the AI service. The second advantage is model independence; the goal is to enable switching to different LLMs while maintaining the same service layer. Another advantage is that security controls can be applied at a central point.

The AI service layer does not merely provide natural language processing functionality; it also acts as a security checkpoint. Comments or requests from the user are cleaned, classified, and masked if necessary within this layer, and then the model is called.

4. Security Approach

In the proposed architecture, security is treated not as a feature added to the system after the fact, but as a fundamental design principle. For this reason, defense mechanisms have been defined at various levels, ranging from the application layer to AI integration, and from traffic management to content sanitization.

4.1 DDoS/DoS Attacks and Rate Limiting

The system is protected against heavy request traffic and botnet-based load-generation attempts using a .NET 8-based rate limiting approach. The server-wide concurrent request capacity has been limited, IP-based request quotas have been defined, and clients exceeding these quotas are tracked in Redis and blacklisted. Since the ban (blocked) information stored in Redis is preserved even if the application is restarted, it provides a more persistent defense compared to temporary memory-based solutions (Figure 2).

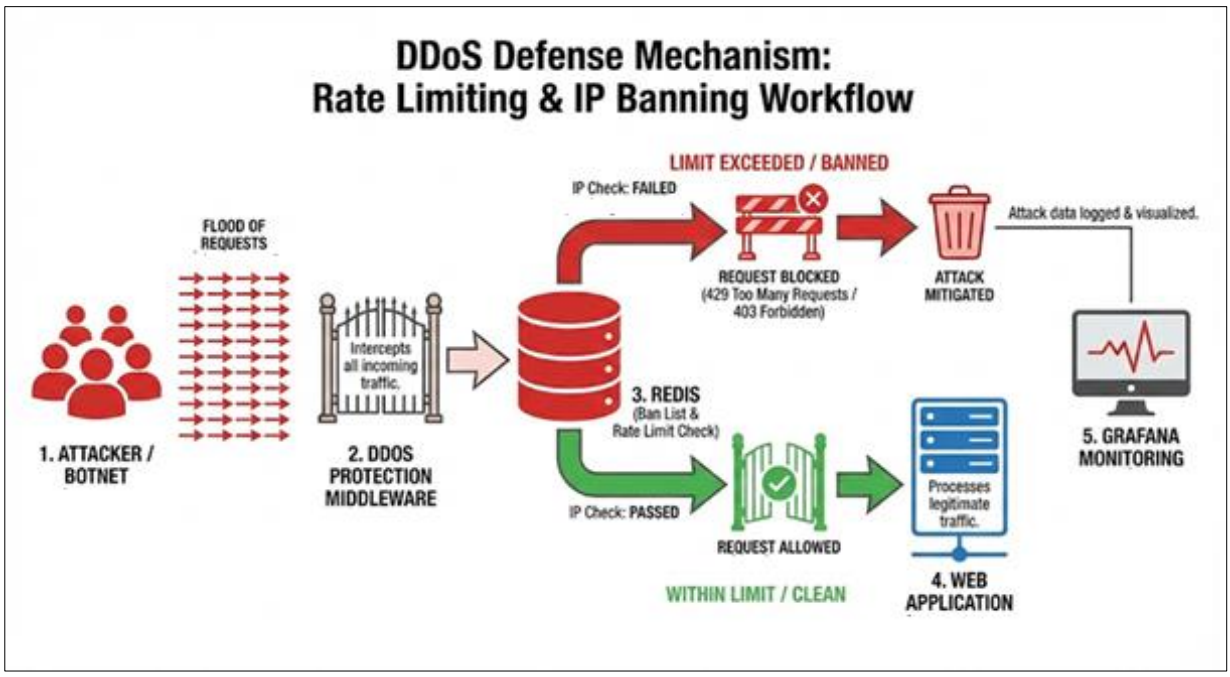


Figure 2. Illustration of DDoS/DoS attacks, rate limiting, and IP blocking

4.2 Detection of Bots and Vulnerability Scanners

User-Agent analysis and error rate monitoring mechanisms have been implemented to counter scraping and vulnerability discovery attempts carried out by automated tools. Requests that do not carry a browser identifier or that carry known automation signatures are rejected. Clients that generate a large number of 404 or 400 errors in a short period of time are labeled as bots or attackers.

4.3 Prompt Injection and Data Privacy

One of the most critical threats to AI services is prompt injection attacks [6]. For this reason, user inputs are examined using regex-based rules defined within SecurityGuard. Patterns intended to bypass system instructions, change roles, or exceed restrictions are blocked before they reach the model.

Additionally, data masking is applied to prevent the transfer of personal data to third-party AI services. Sensitive information such as phone numbers and email addresses is identified in advance and anonymized, thereby ensuring data privacy.

4.4 Content Sanitization and XSS Prevention

User-generated content may contain both data quality issues and malicious code injections [7]. For this reason, the ContentProcessor component is designed as a preprocessing layer that cleans HTML tags, reduces meaningless character repetitions, and filters out spam content. The goal is to both reduce the risk of XSS and improve the quality of the data sent for AI analysis.

5. Observability and Operational Monitoring

In cloud-based systems, simply implementing security mechanisms is not sufficient. It is also necessary to measure the effectiveness of these mechanisms and monitor incidents [8]. Within the scope of this study, real-time observability was achieved by using the Prometheus metrics collection system in conjunction with Grafana dashboards.

Events such as the number of blocked IP addresses, attack types, traffic volume, and error rates can be monitored through custom metrics. This approach helps make firewall logic decisions visible to the operations team and facilitates faster analysis of potential attack scenarios.

6. Conclusion and Discussion

In this study, a microservice-based, security-focused, and observable architecture for AI-powered e-commerce systems is proposed. The proposed architecture offers a multi-tiered solution in which the web application, MCP-based AI service, database, Redis, Prometheus, and Grafana components work together.

The architecture incorporates rate limiting, Redis-based permanent banning, bot and vulnerability scanner detection, prompt injection filtering, data masking, content cleansing, and operational monitoring mechanisms. In this regard, the study presents a practical framework for secure service design in modern e-commerce applications integrated with artificial intelligence.

In addition, the transition from a monolithic architecture to a microservices architecture and the use of Docker-based containers have improved the system's portability and ease of maintenance. Isolating the web application using McpService has also contributed to system resilience by limiting the potential spread of errors.

However, the current study primarily describes the system design and the security components implemented. Consequently, the most significant limitation of the study is that metrics such as security effectiveness, false positive rates, additional latency costs, and impacts on the user experience have not yet been supported by quantitative experiments. Another limitation is that prompt injection detection relies heavily on regex-based rules. While this approach is useful for known patterns, machine learning-supported or policy-based additional control layers may be required against more complex or indirect attacks.

In future work, the proposed architecture could be evaluated through quantitative experiments, and success rates could be measured across different attack scenarios. Additionally, the effects on latency and resource consumption could be analyzed, and comparative performance evaluations could be conducted with different large language models.

References

1. Singh, S., Muntean, C. H., & Gupta, S. (2026). Resilient microservices: an investigation into Istio effectiveness in Kubernetes. Cluster Computing, 29(1), 27. https://doi.org/10.1007/s10586-025-05750-x

2. Şahinaslan, E., & Şahinaslan, Ö. (2022). Database Recovery Technıques in Mıcrosoft Sql Server. International Journal of Innovative Engineering Applications, 6(1), 158-169. https://doi.org/10.46460/ijiea.1070325

3. Chopade, R., & Pachghare, V. (2021). A data recovery technique for Redis using internal dictionary structure. Forensic Science International: Digital Investigation, 38, 301218. https://doi.org/10.1016/j.fsidi.2021.301218

4. Saputra, M. Y. E., Noprianto, Arief, S. N., Wijayaningrum, V. N., & Syaifudin, Y. W. (2024, January). Real-time server monitoring and notification system with Prometheus, Grafana, and Telegram integration. In 2024 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETSIS) (pp. 1808-1813). IEEE. https://doi.org/10.1109/ICETSIS61505.2024.10459488

5. Dewo, K. T., Yasin, V., Budiman, T., Sianipar, A. Z., & Yulianto, A. B. (2023, November). IT Infrastructure Dashboard Monitoring Application Development Using Grafana And Promotheus, a Case Study at Astra Polytechnic School. In 2023 International Conference of Computer Science and Information Technology (ICOSNIKOM) (pp. 1-5). IEEE. https://doi.org/10.1109/ICoSNIKOM60230.2023.10364485

6. Lee, R. W., Jun, T. J., Lee, J. M., Cho, S. I., Park, H. J., & Suh, J. (2025). Vulnerability of Large Language Models to Prompt Injection When Providing Medical Advice. JAMA Network Open, 8(12), e2549963. https://doi.org/10.1001/jamanetworkopen.2025.49963.

7. Duarte, J. D., Cândido, G. D., De Britto Filho, J. R. A., Neto, J. S., Costa, E. J., Da Costa, J. P. J., & De Melo, L. P. (2026). A Systematic Review of Prompt Injection Attacks on Large Language

Models: Trends, Taxonomy, Evaluation, Defenses and Opportunities. IEEE Access. vol. 14, pp. 12875-12899, https://doi.org/10.1109/ACCESS.2026.3656849

8. Keskin, N., Kıran, A. N., Eğdemir, F. K., & Eren, T. (2020). Multicriteria Decision Making Methods and Service Provider Selection to Cloud Information Security Requirements. International Journal of Information Security Engineering, 6(1), 45-60. https://izlik.org/JA69KB64WC