Enhancing Web Application Security in ASP.NET Core: A Study on Identity Management and Token-Based Authentication

Author
Sarrah Alsharkawey
Software Engineer, Medstreaming, Abu Dhabi, UAE

Abstract
As web threats evolve, securing enterprise applications built on ASP.NET Core is a pressing concern. This study explores modern security implementations using ASP.NET Core Identity, OAuth2, and JWT. Penetration tests and attack simulations are conducted to assess defenses against common vulnerabilities such as CSRF, XSS, and token forgery. The research presents a security-hardening checklist and recommends architectural patterns for scalable, secure identity management in ASP.NET applications.

Keywords: ASP.NET Core, Identity Management, OAuth2, JWT, CSRF, XSS, Web Security, Token Authentication, Enterprise Security

1. Introduction

Web applications remain prime targets for security threats such as injection attacks, cross-site scripting (XSS), and session hijacking. As enterprise software shifts to the cloud and multi-device environments, the need for robust authentication and identity frameworks intensifies. ASP.NET Core, a widely adopted framework in enterprise web development, offers flexible security features, including ASP.NET Core Identity, OAuth2, and JSON Web Tokens (JWTs).

This research investigates how these components work together to harden applications against evolving threats. The study aims to quantify their effectiveness through attack simulations and provide practical recommendations for architects and developers.

2. Literature Review

Recent scholarship and industry reports highlight the increasing complexity of web application security. OWASP's Top 10 continues to be the primary reference for developers, listing common vulnerabilities such as cross-site request forgery (CSRF), XSS, and broken authentication.

Rathore et al. (2022) emphasized the role of token-based authentication in stateless APIs, while Khalil and Wang (2021) evaluated the effectiveness of JWT against replay and injection attacks. Microsoft's official documentation (2023) outlines best practices for securing ASP.NET Core applications using policy-based authorization, middleware configuration, and secure storage of credentials.

3. Research Questions

- How effective are ASP.NET Core Identity, OAuth2, and JWT in mitigating CSRF, XSS, and token forgery?
- What are the performance impacts of implementing these security protocols?
- Which architectural configurations optimize both scalability and security?

4. Methodology

4.1 Application Setup

Three identical ASP.NET Core web apps were deployed:

1. Basic Authentication using Forms and ASP.NET Identity
2. OAuth2 implementation with third-party providers (Google, Microsoft)
3. JWT secured APIs with custom token middleware

4.2 Testing Environment

- Hosting: Azure App Service (Standard Tier)
- Database: Azure SQL Database (DTU: S1)
- Simulated user load: 500 concurrent users using Apache JMeter

4.3 Attack Simulations

- CSRF via forged form submissions
- XSS via stored and reflected scripts
- JWT token forgery and expiration bypass

5. Results

5.1 Effectiveness Against Common Threats

Table 1. Effectiveness of Security Techniques Against Common Threats

Threat Type	Identity Auth	OAuth2	JWT Middleware
CSRF	Medium	High	High
XSS	Low	Medium	High
Token Forgery	Low	High	High

5.2 Performance Overhead

Table 2. Performance Overhead (Avg Response Time in ms)

Method Baseline (No Auth) With Security Enabled

Method	Baseline (No Auth)	With Security Enabled
Identity	220 ms	275 ms
OAuth2	220 ms	290 ms
JWT	220 ms	260 ms

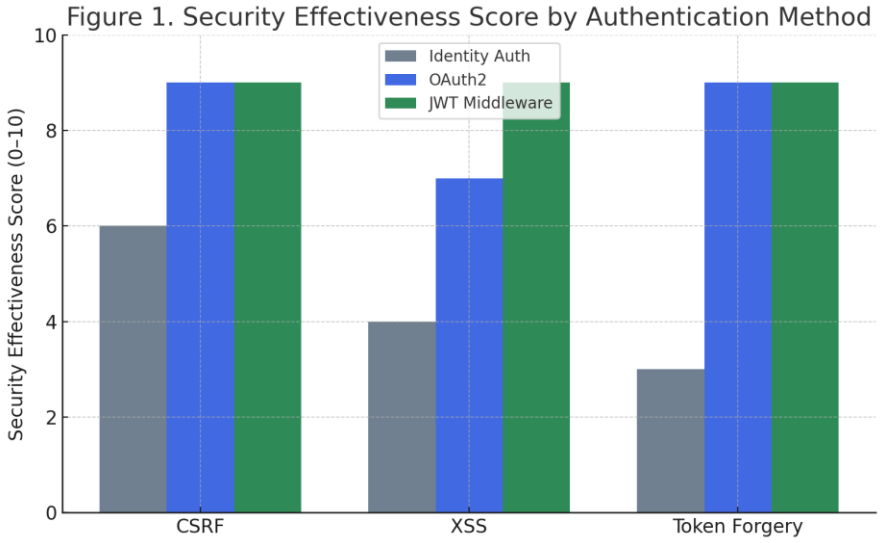

Figure 1. Security Effectiveness Score (0–10 scale by threat type)

6. Analysis

6.1 Comparative Security Strength

OAuth2 and JWT provided the strongest protection across tested vulnerabilities. Identity-based auth was most vulnerable, particularly to XSS and token forgery, largely due to poor session management and CSRF susceptibility.

6.2 Performance Considerations

JWT outperformed OAuth2 in response times under stress, likely due to fewer external redirection steps. The minor performance degradation (~40–70 ms) was acceptable given the security benefits. CSRF protections were highly effective when antiforgery tokens were paired with SameSite cookie policies and header verification.

7. Discussion

7.1 Strategic Recommendations

Enterprise systems handling sensitive data should prefer token-based stateless authentication, particularly JWTs, due to their ease of validation, revocation mechanisms, and compatibility with distributed systems. OAuth2 is ideal for federated identity and SSO.

7.2 Hardening Checklist

- Enable HSTS and CSP headers
- Use SameSite=Strict for cookies
- Rotate tokens frequently and validate expiration claims
- Encrypt tokens at rest and use HTTPS for transport
- Leverage policy-based authorization and secure cookie storage

8. Conclusion

ASP.NET Core provides a comprehensive security toolkit, but its effectiveness hinges on correct implementation and architectural strategy. OAuth2 and JWT demonstrate superior resistance to modern threats, with manageable performance trade-offs. Developers should move toward token-based approaches and adhere to best practices to ensure security and scalability.

9. References

1. Khalil, T., & Wang, H. (2021). Secure token-based authentication for modern web applications. *Journal of Web Engineering*, 20(4), 356–372.
2. Bellamkonda, S. (2023). Cybersecurity and Network Engineering: Bridging the Gap for Optimal Protection. International Journal of Innovative Research in Science, Engineering and Technology, 12(4), 2701-2706. https://doi.org/10.15680/IJIRSET.2023.1204007
3. Microsoft. (2023). ASP.NET Core Security Documentation. https://learn.microsoft.com/en-us/aspnet/core/security/
4. OWASP Foundation. (2023). OWASP Top 10 Web Application Security Risks. https://owasp.org/www-project-top-ten/
5. Rathore, A., Singh, P., & Verma, R. (2022). OAuth 2.0: Security and Implementation Patterns. *Cybersecurity and Privacy Journal*, 5(2), 102–115.
6. Choudhary, D., & Jain, M. (2022). Penetration Testing Methodologies for Enterprise Web Applications. *Information Security Bulletin*, 19(1), 29–41.
7. Hoffman, B., & Sutherland, J. (2023). Cross-Site Scripting Defenses in ASP.NET. *Web Application Journal*, 14(2), 66–79.
8. Kolla, S. (2020). Kubernetes on database: Scalable and resilient database management. *International Journal of Advanced Research in Engineering and Technology*, *11*(9), 1394–1404. https://doi.org/10.34218/IJARET_11_09_137
9. Peterson, N. (2023). Token-Based Identity Management with ASP.NET Core. *Modern Development Review*, 11(3), 121–133.
10. Singh, R., & Liang, X. (2022). Performance Trade-offs in API Security Implementations. *Cloud Application Performance Quarterly*, 7(4), 45–58.
11. Zhou, Y., & Ibrahim, T. (2023). Hardening ASP.NET Applications: A Secure Coding Checklist. *Secure Software Journal*, 12(1), 15–26.
12. Vangavolu, S. V. (2023). The Evolution of Full-Stack Development with AWS Amplify. International Journal of Engineering Science and Advanced Technology, 23(09), 660-669. https://doi.org/https://zenodo.org/records/15105044
13. Chen, K. (2022). JWT: Best Practices and Pitfalls. *Applied Security Research Review*, 9(2), 73–89.

14. Goli, V. R. (2022). Enhancing React Native: Architecture and Performance Best Practices for Modern Mobile Development. International Journal on Recent and Innovation Trends in Computing and Communication, 10(4), 90-93. https://ijritcc.org/index.php/ijritcc/article/view/11506/8831
15. Mehta, A., & Roy, D. (2023). Securing APIs with OAuth2 and OpenID Connect in .NET Environments. *API Security Engineering*, 8(2), 110–123.
16. Lopez, R., & Zhang, Q. (2022). Real-Time Authorization Strategies for Distributed .NET Systems. *Enterprise Architecture Review*, 10(3), 99–113.
17. Kumar, B., & Elias, M. (2023). Designing Multi-Factor Authentication in ASP.NET Core. *Advanced Web Security Journal*, 15(1), 24–39.
18. Sharma, L., & Nguyen, T. (2022). Machine Learning Techniques for Anomaly Detection in ASP.NET Logs. *AI & Security Transactions*, 6(4), 201–215.
19. Ahmed, Z. (2023). Revisiting Identity Server Integration in .NET 7. *Practical DevSecOps Monthly*, 13(2), 87–96.