

Security practices in asp.net core: a gray literature review

Prácticas de seguridad en asp.net core: una revisión de la literatura gris

Práticas de segurança no ASP.NET Core: uma revisão da literatura cinzenta

Paloma Osiris Báez Lara¹
Alejandro Sánchez Marín²
Héctor Xavier Limón Riaño³
María Karen Cortés Verdín⁴
Lizbeth Alejandra Hernández González⁵

Received: March 15th, 2026

Approved: July 15th, 2026

Available: September 1st, 2026

How to cite this article:

P. O. Báez Lara, A. Sánchez Marín, H. X. Limón Riaño, M. K. Cortés Verdín, and L. A. Hernández González, "Security Practices in ASP.NET Core: A Gray Literature Review," *Revista Ingeniería Solidaria*, vol. 22, no. 3, 2026.
doi: <https://doi.org/10.16925/2357-6014.2026.03.03>

Review article. <https://doi.org/10.16925/2357-6014.2026.03.03>

- ¹ Undergraduate Student, Faculty of Statistics and Informatics, Universidad Veracruzana, Mexico.

Email: zs21013871@estudiantes.uv.mx

ORCID: <https://orcid.org/0009-0004-6044-8664>

- ² Undergraduate Student, Faculty of Statistics and Informatics, Universidad Veracruzana, Mexico.

Email: zs21013870@estudiantes.uv.mx

ORCID: <https://orcid.org/0009-0006-8706-4446>

- ³ Researcher, Faculty of Statistics and Informatics, Universidad Veracruzana, Mexico.

Email: hlimon@uv.mx

Corresponding author: hlimon@uv.mx

ORCID: <https://orcid.org/0000-0003-4654-636X>

- ⁴ Researcher, Faculty of Statistics and Informatics, Universidad Veracruzana, Mexico.

Email: kcortes@uv.mx

ORCID: <https://orcid.org/0000-0002-6453-180X>

- ⁵ Researcher, Faculty of Statistics and Informatics, Universidad Veracruzana, Mexico.

Email: lizhernandez@uv.mx

ORCID: <https://orcid.org/0000-0002-8524-3209>



Abstract

Introduction: This article is a product of research focused on the adoption of secure development practices in educational settings, developed at Universidad Veracruzana in 2024. Security is crucial in web development to guarantee data protection against unauthorized access; frameworks like ASP.NET Core offer built-in mechanisms, yet the lack of structured models can lead to system exploitation.

Problem: Despite the availability of technical documentation, knowledge gaps persist regarding the effective contexts for these mechanisms, the specific threats they mitigate, and consolidated practical guidelines.

Objective: This study identifies and analyzes ASP.NET Core security mechanisms, application scenarios, associated threats, and best practices.

Methodology: A Gray Literature Review (GLR) was conducted following Garousi's methodology, analyzing 38 sources from 2022 to 2024 to capture industry practices.

Results: Eight mechanisms and thirteen threats were identified. The review highlights that most vulnerabilities arise from incorrect configurations and improper handling of sensitive data.

Conclusion: These findings provide a foundation for developers to implement secure coding, emphasizing that effectiveness depends on correct configuration rather than just the tools.

Originality: This research bridges the gap between documentation and practitioner experience by linking mechanisms to real-world attack vectors and practices.

Limitations: The scope is limited to ASP.NET Core and gray literature, potentially excluding proprietary industry practices.

Keywords: Secure software development, Web systems, Systematic gray literature review.

Resumen

Introducción: Este artículo es producto de la investigación centrada en la adopción de prácticas de desarrollo seguro en entornos educativos, desarrollada en la Universidad Veracruzana en el año 2024. La seguridad es crucial en el desarrollo web para proteger los datos; marcos como ASP.NET Core ofrecen mecanismos integrados, pero la falta de modelos estructurados puede derivar en la explotación del sistema.

Problema: A pesar de la documentación técnica, persisten brechas de conocimiento sobre los contextos de efectividad de estos mecanismos, las amenazas que mitigan y la falta de guías prácticas consolidadas.

Objetivo: Este estudio identifica y analiza los mecanismos de seguridad de ASP.NET Core, sus escenarios de aplicación, amenazas y mejores prácticas.

Metodología: Se realizó una Revisión de Literatura Gris (GLR) siguiendo la metodología de Garousi, analizando 38 fuentes entre 2022 y 2024.

Resultados: Se identificaron ocho mecanismos y trece amenazas. La revisión resalta que la mayoría de las vulnerabilidades surgen de configuraciones incorrectas y del manejo inadecuado de datos sensibles.

Conclusión: Los hallazgos ofrecen una base para que los desarrolladores implementen codificación segura, enfatizando que la efectividad depende de la configuración correcta más que de las herramientas.

Originalidad: Esta investigación cierra la brecha entre la documentación y la experiencia práctica al vincular mecanismos con vectores de ataque y prácticas reales.

Limitaciones: El alcance se limita a ASP.NET Core y a la literatura gris, pudiendo omitir prácticas propietarias de la industria.

Palabras clave: Desarrollo de software seguro, Sistemas web, Revisión sistemática de literatura gris.

Resumo

Introdução: Este artigo é resultado de uma pesquisa centrada na adoção de práticas de desenvolvimento seguro em ambientes educacionais, desenvolvida na Universidad Veracruzana em 2024. A segurança é fundamental no desenvolvimento web para garantir a proteção dos dados contra acessos não autorizados; estruturas como o ASP.NET Core oferecem mecanismos integrados, mas a ausência de modelos estruturados pode levar à exploração dos sistemas.

Problema: Apesar da disponibilidade de documentação técnica, persistem lacunas de conhecimento sobre os contextos de efetividade desses mecanismos, as ameaças que eles mitigam e a existência de diretrizes práticas consolidadas.

Objetivo: Identificar e analisar os mecanismos de segurança do ASP.NET Core, seus cenários de aplicação, ameaças associadas e boas práticas.

Metodologia: Foi realizada uma Revisão de Literatura Cinzenta (GLR), seguindo a metodologia de Garousi, com análise de 38 fontes publicadas entre 2022 e 2024.

Resultados: Foram identificados oito mecanismos e treze ameaças. A revisão destaca que a maioria das vulnerabilidades decorre de configurações incorretas e do tratamento inadequado de dados sensíveis.

Conclusão: Os resultados oferecem uma base para que os desenvolvedores implementem práticas de programação segura, enfatizando que a eficácia depende mais da configuração correta do que das próprias ferramentas.

Originalidade: A pesquisa estabelece uma conexão entre a documentação e a experiência dos profissionais, relacionando mecanismos a vetores de ataque e práticas do mundo real.

Limitações: O estudo está limitado ao ASP.NET Core e à literatura cinzenta, podendo excluir práticas proprietárias da indústria.

Palavras-chave: Desenvolvimento seguro de software; sistemas web; revisão sistemática de literatura cinzenta.

1. INTRODUCTION

Software development is a fundamental process for creating applications and systems that meet user requirements and operate effectively across different environments [1]. The dynamic nature of requirements, influenced by rapid technological changes, contributes to uncertainty and complexity [2, 3]. This complexity, combined with the increase in attack vectors and opportunities for malicious actors, has made software security an increasingly critical concern. In our modern landscape, data security has become an invaluable asset for organizations, highlighting its importance. However, safeguarding this information represents a significant challenge, as its management involves numerous actors and complex factors that complicate effective control and protection [4, 5, 6].

Given the need for software to meet specific quality requirements, such as security, developers often utilize development frameworks to help address these challenges. Such frameworks can help improve development efficiency and support

reliability efforts while also providing structured approaches to manage the increasing complexity of web applications [7]. In terms of security specifically, frameworks often come with secure default configurations, which can help developers avoid common security pitfalls. Additionally, software development frameworks facilitate the process by offering pre-built modules and reusable components, which can reduce some of the time and resources typically required for project execution [7, 8].

Frameworks, such as ASP.NET Core, integrate security mechanisms within their tool suite; however, many developers lack the necessary skills to effectively use **these** mechanisms [9, 10]. As a result, this skill gap becomes a critical concern, as software development frameworks are inherently complex, and their effectiveness largely depends on the quality of development practices [11, 12]. A lack of knowledge often leads to bad practices, which in turn can introduce security vulnerabilities [13]. In contrast, following best practices helps mitigate complexity, promoting code clarity and maintainability [14, 15].

This Gray Literature Review (GLR) focuses on the ASP.NET Core web development framework, with the aim of examining its built-in security mechanisms and their potential to improve the security of web systems. Our interest in this framework arises from its prevalent use among students in our ongoing research on the adoption of secure development practices in educational settings. By analyzing ASP.NET Core in depth, we seek to identify the security mechanisms it offers and assess how these can be leveraged to enhance the understanding and application of secure coding principles among students in software-related programs.

Despite the availability of technical documentation for ASP.NET Core, significant knowledge gaps remain regarding the security mechanisms it provides. It is often unclear which mechanisms are available, in what contexts they are most effective, and which specific threats they are designed to mitigate. Furthermore, there is a lack of sources that identify the best and bad practices related to secure system development using this framework, as well as the common challenges developers face when applying its security features.

The objective of this **Gray Literature Review (GLR)** is to examine the security features provided by ASP.NET Core. This includes identifying the security mechanisms integrated into the framework, the context in which each is applied, the threats and vulnerabilities they aim to mitigate, as well as the best and bad practices for their use. We chose to focus exclusively on gray literature, as we found little relevant information on this topic in academic sources. Furthermore, the inclusion of gray literature has been recognized as a vital practice in software engineering research to aggregate available evidence and mitigate publication bias [16].

The results of this review provide a current perspective on the security aspects provided by ASP.NET Core. Furthermore, our findings can serve as a valuable resource for developing educational materials on web application security using ASP.NET Core, helping to raise awareness and strengthen security knowledge among junior developers and students.

This paper is structured as follows: Section 2 describes the methodology employed to carry out this gray literature review. Section 3 presents the results obtained, including answers to the research questions. Section 4 provides a discussion of the findings, while Section 5 addresses possible threats to the validity of the study. Finally, Section 6 presents the general conclusions and proposes lines of future work.

2. METHODOLOGY

We conducted a gray literature review following the methodology proposed by Garousi et al. [17]. This approach is designed for multivocal literature reviews, which incorporate both white and gray literature. The methodology comprises three main phases: review planning, review conduct, and review reporting. The following sections detail the activities carried out in each phase.

2.1 Identification of the need for gray literature

As a first step, we evaluated the appropriate literature type for this review using the checklist proposed by [17], which is presented in Table 1. This checklist consists of seven questions designed to help determine the relevance of gray literature.

Table 1. Questions to decide whether to include gray literature.

ID	Question	Possible answer (Yes / No)
1	Is the subject “complex” and not solvable by considering only the formal literature?	Yes
2	Is there a lack of volume or quality of evidence, or a lack of consensus of outcome measurement in formal literature?	Yes
3	Is the contextual information important to the subject under study?	Yes
4	Is it the goal to validate or corroborate scientific outcomes with practical experiences?	Yes
5	Is it the goal to challenge assumptions or falsify results from practice using academic research or vice versa?	No

(continúa)

(viene)

ID	Question	Possible answer (Yes / No)
6	Would a synthesis of insights and evidence from the industrial and academic community be useful to one or even both communities?	Yes
7	Is there a large volume of practitioner sources indicating high practitioner interest in a topic?	No

Source: own work

Since five of the questions were answered affirmatively and a preliminary review of the white literature yielded no significant findings, we decided to focus exclusively on gray literature.

2.2 Research questions

We based our gray literature review on six research questions:

- **RQ1:** What security mechanisms does ASP.NET Core provide?
- **RQ2:** In what possible contexts or scenarios are ASP.NET Core security mechanisms used?
- **RQ3:** What threats and/or vulnerabilities are associated with each of the security mechanisms of ASP.NET Core?
- **RQ4:** What best practices are associated with security mechanisms of ASP.NET Core?
- **RQ5:** What bad practices are associated with security mechanisms of ASP.NET Core?
- **RQ6:** What are the challenges of using the security mechanisms identified for ASP.NET Core?

2.3 Search process

Our initial research was based on an article from the official documentation of the ASP.NET Core framework [18]. Using this documentation, we identified key terms. As a result, we created nine search strings: one general query covering all security mechanisms and eight more specific queries.

The general search string was limited to selected websites, identified through a prior empirical evaluation, which helped us determine which domains—primarily blogs and technical documentation—provided the most relevant and useful results for the purposes of this review. The general query is as follows:

General search string (targeting all mechanisms and limited to selected sites):

"ASP.NET Core" AND (Feature OR Mechanisms) AND Security AND (Good Practices OR Bad Practices OR "Anti-Pattern") AND (Authentication OR Authorization OR "Data protection" OR HTTPS Enforcement OR App Secrets Management OR CSRF OR CORS OR XSS) AND (site:medium.com OR site:microsoft.com OR site:linkedin.com)

To complement the general search string, we created a set of more focused search strings, which share a common structure, beginning with general terms but each ending with a term tailored to a particular security mechanism or threat.

Specific search strings (same base, varied by focus):

"ASP.NET Core" AND Security AND

- Authentication
- Authorization
- Data Protection
- HTTPS Enforcement OR HTTP
- Cross-Site Scripting OR XSS
- Cross-Site Request Forgery OR CSRF OR XSRF
- CORS
- Safe Storage of App Secrets OR Secrets Management OR App Secrets Security AND Security

It is important to note that some of these specific strings focus on security threats rather

than mechanisms. We adopted this strategy based on the observation that certain security mechanisms are more commonly discussed in relation to the threats they mitigate, which makes threat-based searches more effective for retrieving relevant information.

2.4 Selection process

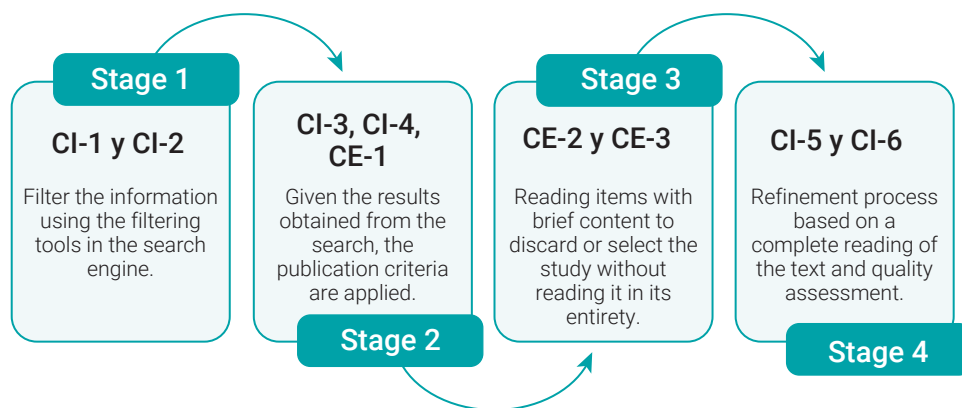
To identify relevant sources, we applied specific inclusion (IC) and exclusion (EC) criteria, divided into 4 stages. Additionally, due to the nature of gray literature, we established stopping criteria (SC). Table 2 details the criteria applied.

Table 2. Inclusion, exclusion, and stopping criteria.

Criteria	Description
IC-1	Results published between 2022 and 2024 or releases made from ASP.NET Core versions 7.0 and 8.0.
IC-2	The source was originally written in English.
IC-3	The name of the author or organization can be identified in the source.
IC-4	The title indicates that it is related to at least one research question.
IC-5	After reading the source, it should be confirmed that it answers one or more of the research questions.
IC-6	The source meets the criteria of the quality assurance evaluation for gray literature.
EC-1	The source does not contain a release date, nor does it specify the versions of the framework.
EC-2	The source is not a blog and belongs to the third level of gray literature and strictly should not have as its objective the promotion or sale of any type of tool, service, or job vacancy.
EC-3	The source is duplicated.
SC-1	Theoretical saturation: The results do not present new concepts after analyzing ten consecutive results that no longer provide relevant information.
SC-2	Limited effort: 200 results have been reached.

Source: own work

The process for selecting sources is structured into four stages, guided by inclusion and exclusion criteria, with the aim of conducting a systematic search that filters out irrelevant material while retaining pertinent sources. This is shown in Figure 1.

**Figure 1.** Selection process.

Source: own work

The search and selection process was conducted in incognito mode using Google Search to minimize bias from cookies and browsing history, ensuring more impartial results.

Figure 2 presents the total number of resources retrieved at each stage of the search strings. In total, 38 resources were ultimately selected.

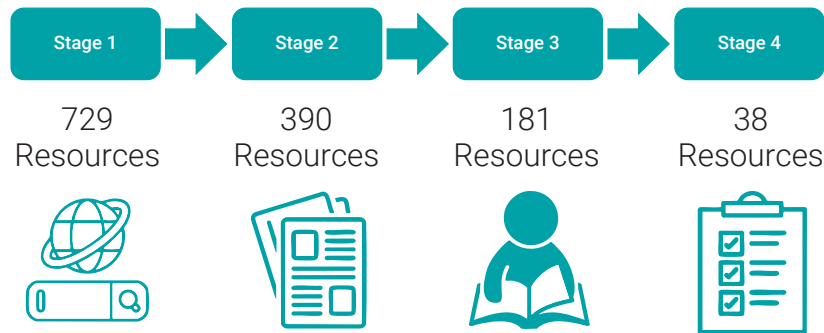


Figure 2. Results of the selection process.

Source: own work

2.5 Quality assessment

A quality assessment was conducted using an adaptation of the checklist by [17]. Table 3 shows the checklist, which has a maximum score of 6.5. Each criterion was rated on a scale from 0 (No) to 1 (Yes). Furthermore, a bonus of 1.5 points was awarded to sources that provided greater depth, ensuring that only those offering relevant information met the quality standards. Considering these adjustments, a minimum score of 4 was required for a source to be considered acceptable.

Table 3. Source Quality Criteria.

Criteria	Question
Authority of the producer	1.1 Does the author have demonstrable experience or a degree in security or software development?
1. Is the source of the resource reliable and recognized in the area?	1.2 Is the organization or company recognized for its expertise or authority in security or software development?
Objectivity	2. Is the resource objective, or is it based merely on opinion?
Related sources	3. Has the publication been referenced by other resources?
Subject depth	4. Does the resource provide in-depth analysis of the topic, or does it offer only superficial treatment?
Type of source	<ul style="list-style-type: none"> • Book, journal, thesis, government report, white paper (1 point). • Annual report, news article, question/answer sites, wiki article, blog (0.5 points).

Source: own work

This evaluation process led to the exclusion of several sources: few achieved high quality (6 to 6.5 points), most obtained acceptable quality (4 points), and some were considered good quality (5 to 5.5 points). As a result, 19 sources obtained acceptable quality, 12 sources good quality, and only 7 sources high quality. This is illustrated in Figure 3.

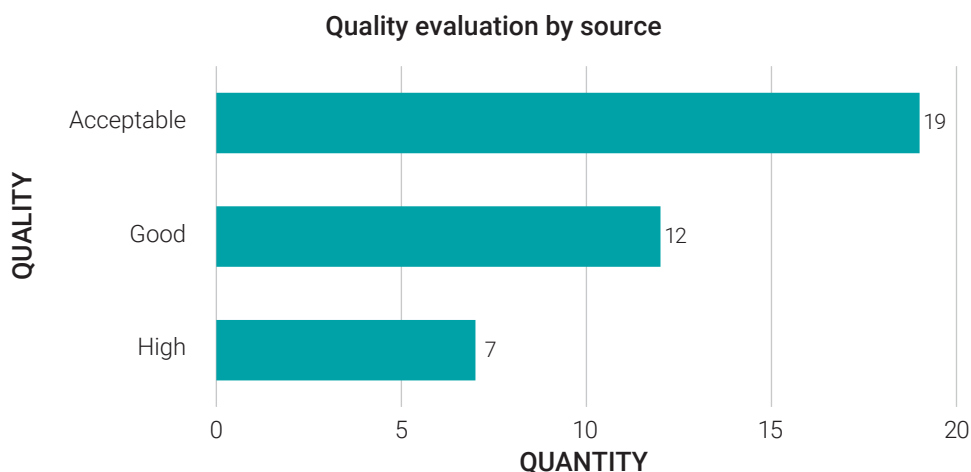


Figure 3. Quality evaluation of selected sources.

Source: own work

2.6 Data extraction and synthesis process

For the extraction process, a structured format was established to organize and systematize the collected information. This format comprises two main sections: one designed to record general data for each source (such as title, authors, year, publication type, and reference), and another focused on capturing content relevant to the research questions.

The second section includes aspects such as the security mechanisms offered by ASP.NET Core, usage contexts, associated threats or vulnerabilities, best and bad practices, as well as implementation challenges. Due to space constraints, the specific format is not presented in this document.

To integrate the collected data, a narrative synthesis was conducted based on the methodological framework proposed by Popay et al. [19]. This approach enables the review and integration of findings from multiple studies through detailed textual descriptions.

The process involved the following steps:

- Development of a theoretical model: **to** describe how and why the studies included in the review are expected to answer the research questions.
- Development of **a** preliminary synthesis: **to** organize the information **and** describe patterns in the studies.

The final synthesis was constructed using textual descriptions, tables, and the developed theoretical model.

3. RESULTS

3.1 Distribution of results by year

Most of the selected sources were published in 2024, accounting for 68.42% of the total.

This was followed by publications from 2023 (26.32%) and, to a lesser extent, those from 2022 (5.26%). All sources are **from** the last three years, which aligns with the framework version examined in this literature review. A more detailed breakdown is shown in Figure 4.

In the following sections, we present the results corresponding to each research question.

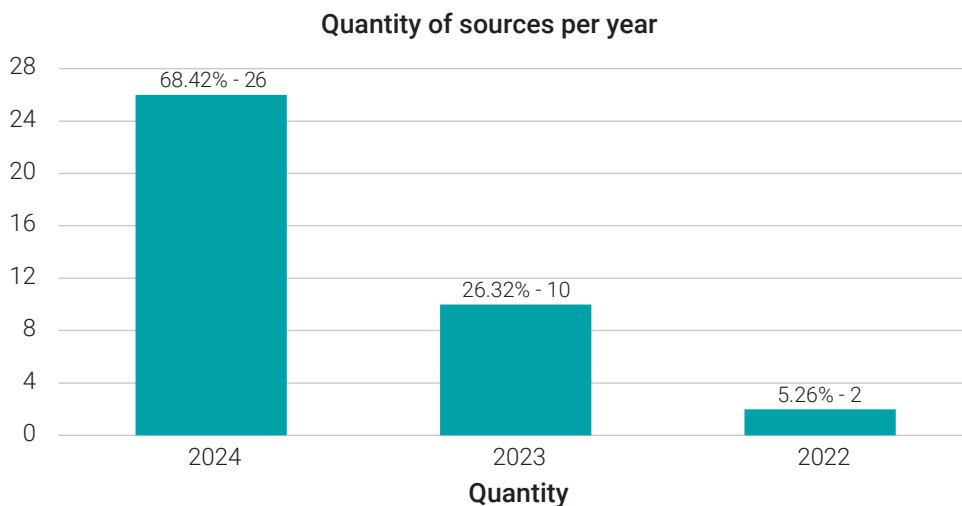


Figure 4. Percentage of sources per year.

Source: own work

3.2 RQ1. What security mechanisms does ASP.NET Core provide?

Authentication: Is the process by which an identity is verified to ensure that it is authentic. It is performed by comparing credentials that are stored in a database, application, resource, or operating system [18].

Authorization: According to [18], it refers to the process of determining which actions or resources a user can access after having been successfully authenticated. Once authenticated, the user receives an identity that is usually used to perform authorization checks.

Data protection: A mechanism to encrypt and protect sensitive information such as cookies, session tokens, and confidential user data. ASP.NET Core provides an API for this implementation called the Data Protection API [20].

HTTPS Enforcement: Ensures that all communication between the client and the server is encrypted by using the SSL/TLS protocol [21].

Secrets Management: Allows the storage of sensitive information such as API keys, passwords, or database connection strings outside the application source code. In software development, the use of this mechanism makes it easier to keep confidential information protected and separate from the code, avoiding the risk of exposure [47].

Anti-forgery tokens: A security mechanism provided by ASP.NET Core to prevent Cross-Site Request Forgery (CSRF) attacks. It is a unique token that is assigned to each user session and is included in a hidden field in HTML forms [22]. When form data is sent to the server, the token is validated to ensure that the request originates from a legitimate client [23]. If the token is not included or does not match the stored value, the server rejects the potentially malicious request [22].

Cross-Origin Resource Sharing (CORS): Enables web applications to control which external domains can access their resources; i.e., it regulates communications from domains that are different from the system domain [24].

HTML Encoder: Ensures that the data returned by the API is correctly encoded to prevent it from being executed as malicious code in the user's browser [25]. ASP.NET Core incorporates the Razor framework, which automatically encodes variables to prevent the execution of malicious scripts [26].

3.3 RQ2. In what possible contexts or scenarios are ASP.NET Core security mechanisms used?

Table 4 synthesizes the main application contexts in ASP.NET Core and schematically presents the scenarios where they are most suitable.

Table 4 . Usage scenarios for security mechanisms in ASP.NET Core.

Mechanism	Contexts or usage scenarios
Authorization	<ul style="list-style-type: none"> • Role-based: simple applications with limited permissions [27]. • Claim-based: complex applications with multiple roles and granular control [27]. • Implementation in financial applications [28], CMS [29], e-commerce [29], enterprise applications [29], and social networks [29].
Authentication	<ul style="list-style-type: none"> • Windows: intranet applications. • Forms: users with their own credentials. • OAuth, OpenID Connect, SAML: modern and scalable applications with external providers [27].
Secrets Management	<ul style="list-style-type: none"> • Scenarios where storing passwords or sensitive keys are required, reducing exposure risks [30].
Anti-forgery tokens	<ul style="list-style-type: none"> • Prevention of CSRF attacks on sensitive sites, such as banking platforms, validating tokens in each request [31].
HTML Encoder	<ul style="list-style-type: none"> • Applications with user inputs (e.g., comments), preventing script execution and XSS attacks [32].
Data Protection	<ul style="list-style-type: none"> • Financial sectors where the protection of sensitive banking and personal information is crucial to maintain data confidentiality [33].

Source: own work

3.4 RQ3. What threats and/or vulnerabilities are associated with each of the security mechanisms of ASP.NET Core?

From the selected sources, diverse threats and vulnerabilities (TV) impacting the use of security mechanisms were identified, totaling 13 findings. The results are presented in Table 5.

Table 5. Threats and vulnerabilities.

Threat / Vulnerability (TV)	Description
TV-01. Cross-site Scripting (XSS)	Attack in which an attacker injects malicious scripts into a trusted website [27].
TV-02. Data injection	Attackers inject untrusted content such as malicious code, styles, or images [34].
TV-03. Clickjacking	A malicious website embeds a legitimate application inside an invisible or disguised iframe [34].
TV-04. Mixed content	An HTTPS page starts loading resources over HTTP, which exposes the application to a possible attack [34].
TV-05. Cross-site Request Forgery (CSRF)	Attack in which an attacker tricks a user into performing unwanted actions while authenticated [22].

(continúa)

(viene)

Threat / Vulnerability (TV)	Description
TV-06. Unauthorized Access	Malicious users attempt to access sensitive data or perform actions without authorization, which can lead to data breaches [36].
TV-07. Insecure Direct Object Reference (IDOR)	The application directly exposes references to internal objects (such as file paths or keys) without validating the appropriate access rights [27].
TV-08. SQL Injection	Unauthorized users inject malicious SQL code that is executed in the database management system [35].
TV-09. Data tampering	Data transmitted between the client and the API can be intercepted and modified [36].
TV-10. URL tampering	Access to restricted resources by modifying the URL to bypass access control [37].
TV-11. Privilege escalation	A user obtains higher privileges than those originally assigned in application [37].
TV-12. Token hijacking or manipulation	Attack that exploits weaknesses in security tokens (e.g., JWT) to access restricted data or actions [37].
TV-13. Man-in-the-Middle Attack	An attacker intercepts communication between the user and the server, injecting malicious code [34].

Source: own work

Table 6 shows the relationship between each threat and vulnerability (TV) and the identified security mechanisms. The results highlight that HTML Encoders and Authorization are the most implemented security mechanisms, each associated with four vulnerabilities, reflecting their relevance in mitigating multiple security risks. Likewise, Authentication, Data Protection, and HTTPS Enforcement appear as frequently used mechanisms that contribute to addressing various threats.

Table 6. Security mechanisms are associated with threats or vulnerabilities.

Threat or vulnerability / Security Mechanism	Authentication	Authorization	Data Protection	HTTPS	Anti-forgery tokens	Secrets Management	CORS	HTML Encoder
TV-01. Cross-site Scripting (XSS) [27].								X
TV-02. Data injection [34].								X
TV-03. Clickjacking [34].								X
TV-04. Mixed content [34].				X				X
TV-05. Cross-site Request Forgery (CSRF) [22].					X		X	
TV-06. Unauthorized Access [36].	X	X	X			X		
TV-07. Insecure Direct Object Reference (IDOR) [27].		X						
TV-08. SQL Injection [35].						X		
TV-09. Data tampering [36].			X	X				

(continúa)

(viene)

Threat or vulnerability / Security Mechanism	Authentication	Authorization	Data Protection	HTTPS	Anti-forgery tokens	Secrets Management	CORS	HTML Encoder
TV-10. URL tampering [37].		X						
TV-11. Privilege escalation [37].	X	X						
TV-12. Token hijacking or manipulation [37].	X		X					
TV-13. Man-in-the-Middle Attack [34].				X				

Source: own work

3.5 RQ4. What best practices are associated with the security mechanisms of ASP.NET Core?

The effective implementation of security mechanisms in ASP.NET Core requires the adoption of best practices that reinforce the protection of developed systems. From the selected sources, specific strategies were identified for each security mechanism.

In the case of Authentication, the use of Two-Factor Authentication (2FA) and Multi-Factor Authentication (MFA) is recommended [28]. Likewise, input validation and sanitization [38], the use of JSON Web Tokens (JWT) with strong signature algorithms (such as RS256) [20], validating the token's signature and integrity before processing requests [39], and the implementation of OAuth 2.0 and OpenID Connect significantly strengthen the authentication process [40]. To further protect authentication endpoints from brute-force and Denial-of-Service (DoS) attacks, it is essential to implement rate limiting and restrict request payload sizes [41]. Furthermore, implementing error logging and monitoring to track authentication failures is crucial to proactively detect and respond to security incidents [42].

For Authorization, the use of ASP.NET Core Identity [40], Role-Based Access Control (RBAC) [43], and access permission validation **allows** ensuring that users can only access strictly necessary resources [27]. Furthermore, direct role policies offer greater flexibility than the RequireRole method, as they allow the creation of policies that require multiple roles or other conditions, making the authorization logic easier to maintain [44]. Additionally, encrypting configuration files [23] and using CORS provide an extra layer of security that helps prevent unauthorized access [21].

Regarding Data Protection, it is suggested to employ both symmetric and asymmetric encryption to safeguard user credentials [27], a practice equally applicable to Secrets Management. It is recommended to store secrets **on** secure platforms such

as Azure Key Vault, applying periodic secret rotation [45], and to perform regular code analysis to identify embedded secrets and avoid their accidental exposure [45].

Regarding the implementation of HTTPS, it is advised to use HSTS to enforce secure communications [46], as well as SSL/TLS, which **reinforce** the application by encrypting data in transit and **help** prevent data theft [46]. As an additional security layer, the application of secure HTTP response headers is also recommended [27].

The creation of Content Security Policies (CSP) makes it difficult for attackers to inject malicious content [22], constituting the best practice associated with the use of the HTML Encoder. In turn, the validation of input requests contributes to minimizing the risk of accepting malicious data [22].

Regarding CORS, the importance of proper header configuration is highlighted [24], as well as **the correct order of** calls to ensure the proper application of policies [48]. On the other hand, regarding Anti-forgery tokens, it is recommended to update them after the user has authenticated, redirecting them to a view or Razor page [31].

In summary, effective security in ASP.NET Core requires a comprehensive approach that combines multiple best practices through the protection mechanisms offered by this software development framework.

3.6 RQ5. What bad practices are associated with the security mechanisms of ASP.NET Core?

Several bad practices linked to the security mechanisms offered by ASP.NET Core were identified. However, due to the limited information available in some sources, it was possible to determine these practices in only six of the eight analyzed mechanisms.

Among the bad practices, it was possible to identify that, for Authentication, the use of the Resource Owner Password Credentials Grant (ROPC) constitutes a bad practice, as it exposes passwords on the client side and should only be used in exceptional cases [18]. Likewise, not deleting authentication cookies upon logging out compromises system security [35]. Similarly, exposing sensitive information in error messages can provide attackers with valuable clues to breach the application [28].

Regarding the Authorization mechanism, not applying access denial by default allows improper access, and incorrect configuration of HTTP methods can leave the API exposed to unauthorized users [37].

Regarding Secrets Management, storing sensitive information directly in the source code (hardcoding) or in configuration files [45], as well as using weak keys or

reusing credentials across different environments, significantly increases the risk of vulnerabilities [47].

In the use of HTTPS, using HTTP Strict Transport Security (HSTS) in development environments can generate errors, while the use of self-signed certificates in production environments **facilitates** phishing attacks [49].

In the CORS security mechanism, configuring **AllowAnyOrigin** together with **AllowCredentials** enables the possibility of Cross-Site Request Forgery (CSRF) attacks [30]. Likewise, the use of wildcards and the combination of middleware with attributes such as **[EnableCors] are** risky [24].

Finally, in the case of HTML Encoder, injecting untrusted data into HTML without proper encoding constitutes a bad practice that facilitates Cross-Site Scripting (XSS) attacks. It is also dangerous to concatenate untrusted data in JavaScript or to rely solely on validation without applying encoding [50].

Many of the identified bad practices stem from incorrect configurations or improper handling of sensitive information. Identifying these practices is fundamental, as it allows developers to adopt proactive measures to strengthen system protection and security.

3.7 RQ6. What are the challenges of using the security mechanisms identified for ASP.NET Core?

The implementation of security mechanisms brings advantages in terms of strengthening applications against possible threats. However, there are security mechanisms that present challenges **for implementation**, for example:

CORS can represent a significant challenge due to the difficulty **of** debugging; when errors related to CORS occur, they may not be reported by browsers, making error detection complex. Similarly, **this** security mechanism can become complex in cases where the application needs to support multiple origins that have different permissions, and an overly permissive configuration can expose the application to security risks [48].

For Secrets Management, a challenge is the configuration involved in its use in Visual Studio, since, although this IDE provides a more integrated and user-friendly approach, it requires additional configuration and may have compatibility issues with older or unsupported versions of .NET [47].

Regarding the HTML Encoder, challenges associated with the use of Content Security Policies (CSP) were identified, given that these policies can block legitimate resources (scripts, fonts, styles, or images), thus requiring rigorous configuration.

However, the debugging process is a time-consuming activity and can be overwhelming when the application grows and third-party services or libraries are integrated. Likewise, it was found that not all browsers support the use of CSP; therefore, development is limited to those browsers that are compatible [34].

It is possible to identify that the main challenge presented in **the** security mechanisms is configuration. This is mainly due to the complexity involved in adequately adapting security mechanisms to the different scenarios that arise.

4. DISCUSSION

The results obtained help **to** understand the use and implementation of security mechanisms in web system development with ASP.NET Core. This framework provides a variety of mechanisms (RQ1), each varying in applicability and context (RQ2), which means that the adoption of specific security mechanisms depends on external factors such as project requirements. The findings also reveal that certain vulnerabilities can be associated with security mechanisms, that is, those security mechanisms that help prevent specific vulnerabilities (RQ3). These vulnerabilities align with the study by Aborujilah et al. [51], who reported vulnerabilities identified in ASP.NET Core and prevention methods. However, the omission of validations or incorrect configuration of security mechanisms (RQ6) introduces the risk of attacks, which are caused by bad practices (RQ5). Therefore, best practices (RQ4) play a crucial role in the implementation of security mechanisms and significantly impact the robustness of developed web systems. This study can serve as a basis for raising developers' awareness of the importance of adopting a preventive approach **to** security. However, implementing these recommendations may present challenges; therefore, it is necessary to consider all the information provided to achieve an effective implementation of security mechanisms. Ultimately, security depends not only on the tools used but **also** on how developers understand and apply them.

5. THREATS TO VALIDITY

One threat to the validity of this study is the potential bias in the process of searching for and selecting resources, given that we prioritized certain sources that, after an empirical process, we identified as those that yielded the best results. We mitigated this risk **by using** specific search strings, as these do not define sources, allowing us to obtain varied results and not only from the empirically selected sources.

Another threat to validity arises from the fact that, **during** the assessment, most resources that passed the assessment reached the minimum score, potentially giving rise to unreliable information. However, as a mitigation measure, we ensured that those resources with low scores addressed the topic with sufficient depth to contribute to the research.

The last threat to validity refers to the absence of a methodology specifically designed for gray literature, because we could not find specific methodologies for this type of literature. To address this risk, we followed a similar methodology proposed by Garousi et al. [17] for multivocal literature reviews, which includes gray literature.

6. CONCLUSION

The objective of this work was to identify and analyze the security mechanisms offered by the ASP.NET Core framework. Among the findings, we identified eight security mechanisms that can be considered the most relevant, as they are indicated in Microsoft's official documentation [18]. Additionally, the general results of this work are as follows:

- For the scenarios, although it was not possible to identify them for all mechanisms, we described contexts in which authentication, authorization, secrets management, data protection, HTML Encoder, and anti-forgery tokens can be applied.
- We identified best practices mainly related to authentication and authorization, such as the use of the ASP.NET Core Identity library. Regarding bad practices, it can be highlighted that these were due to incorrect configurations in the mechanisms and inadequate handling of sensitive data.
- Regarding security threats, we associated threats with each security mechanism, but those that stood out most in the literature were XSS, CSRF, and SQL injection attacks.
- Finally, regarding challenges, we found that the greatest difficulties developers face **are** due to performing correct configurations.

In conclusion, this work may serve as a basis for understanding ASP.NET Core security mechanisms and for considering the importance of security in web applications. It is important to highlight that there was limited information in **the** white literature on this topic, so we had to focus our research on gray literature.

For future work, we suggest expanding the practical cases where security mechanisms can be applied to identify a greater number of best and bad practices associated with these mechanisms. Additionally, this gray literature research is intended to serve as the foundation for developing an educational resource, **namely**, a practical guide to ASP.NET Core security, aimed at reinforcing web application security knowledge among junior developers and students.

REFERENCES

- [1] I. Schaefer *et al.*, “Software diversity: State of the art and perspectives,” *International Journal on Software Tools for Technology Transfer*, vol. 14, no. 5, pp. 477–495, Jul. 2012, doi: <https://doi.org/10.1007/s10009-012-0253-y>.
- [2] H. Sadia, S. Q. Abbas, and M. Faisal, “A Bayesian network-based software requirement complexity prediction model,” in *Intelligent Technologies and Applications*. Cham, Switzerland: Springer, 2022, pp. 197–213, doi: https://doi.org/10.1007/978-981-19-3015-7_15.
- [3] M. Haleem, M. F. Farooqui, and M. Faisal, “Tackling requirements uncertainty in software projects: A cognitive approach,” *International Journal of Cognitive Computing in Engineering*, vol. 2, pp. 180–190, 2021, doi: <https://doi.org/10.1016/j.ijcce.2021.10.003>.
- [4] J. C. Najar Pacheco, J. A. Bohada Jaime, and H. C. I. Alemán Novoa, “Vulnerability of information on the Internet of Things,” *Revista Ingeniería Solidaria*, vol. 18, no. 3, pp. 1–17, Sep. 2022, doi: <https://doi.org/10.16925/2357-6014.2022.03.07>.
- [5] N. Mohamed, “Artificial intelligence and machine learning in cybersecurity: A deep dive into state-of-the-art techniques and future paradigms,” *Knowledge and Information Systems*, vol. 67, no. 8, pp. 6969–7055, 2025, doi: <https://doi.org/10.1007/s10115-025-02429-y>.
- [6] K. Svensson, D. Axelrod, M. Mohamad, and R. Wohlrab, “Guidelines for supporting software engineers in developing secure web applications,” in *Product-Focused Software Process Improvement (PROFES2024), Lecture Notes in Computer Science*, vol. 15452. Cham, Switzerland: Springer, 2024, pp. 123–138, doi: https://doi.org/10.1007/978-3-031-78386-9_9.
- [7] Y. R. Selivorstova, I. V. Liutenko, and S. V. Orekhov, “Frameworks analysis and evaluation used in web application development,” *Bulletin of National Technical University “KhPI”. Series: System Analysis, Control and Information Technologies*, no. 2, pp. 44–50, Dec. 2020, doi: <https://doi.org/10.20998/2079-0023.2020.02.08>.

- [8] M. del Pilar Salas-Zárate, G. Alor-Hernández, R. Valencia-García, L. Rodríguez-Mazahua, A. Rodríguez-González, and J. L. López-Cuadrado, "Analyzing best practices on web development frameworks: The Lift approach," *Science of Computer Programming*, vol. 102, pp. 1–19, May 2015, doi: <https://doi.org/10.1016/j.scico.2014.12.004>.
- [9] S. Bangani, L. Futcher, and J. van Niekerk, "An approach to teaching secure programming in the .NET environment," in *ICT Education, Communications in Computer and Information Science*, vol. 1136. Cham, Switzerland: Springer, 2019, pp. 35–49, doi: https://doi.org/10.1007/978-3-030-35629-3_3.
- [10] A. Alkussayer and W. H. Allen, "The ISDF framework: Integrating security patterns and best practices," in *Advances in Information Security and Its Application (ISA 2009), Communications in Computer and Information Science*, vol. 36. Berlin, Germany: Springer, 2009, pp. 17–28, doi: https://doi.org/10.1007/978-3-642-02633-1_3.
- [11] S. A. Slaughter, "Software development practices and software maintenance performance: A field study," Ph.D. dissertation, (ND), USA, 1996, doi: <https://doi.org/10.1287/mnsc.44.4.433>.
- [12] S. Maheshwari, "Mapping challenges in teaching-learning software testing," in *Proc. IEEE Frontiers in Education Conf. (FIE)*, Cincinnati, OH, USA, Oct. 2019, pp. 1–9, doi: <https://doi.org/10.1109/FIE43999.2019.9028452>.
- [13] T. E. Gasiba, U. Lechner, M. Pinto-Albuquerque, and D. Mendez, "Is secure coding education in the industry needed? An investigation through a large-scale survey," in *Proc. 43rd International Conference on Software Engineering: Joint Track on Software Engineering Education and Training (ICSE-JSEET)*, May 2021, pp. 241–252, doi: <https://doi.org/10.1109/ICSE-SEET52601.2021.00034>.
- [14] R. D. Banker, G. B. Davis, and S. A. Slaughter, "Software development practices, software complexity, and software maintenance performance: A field study," *Management Science*, vol. 44, no. 4, pp. 433–450, 1998, doi: <https://doi.org/10.1287/mnsc.44.4.433>.
- [15] S. S. Alqahtani, "Beyond the code: Analyzing OSS developers' security awareness and practices," *International Journal of Information Security*, vol. 24, no. 3, (ND), 2025, doi: <https://doi.org/10.1007/s10207-025-01023-1>.
- [16] R. Fatima, A. Yasin, L. Liu, and J. Wang, "The use of grey literature and Google Scholar in software engineering systematic literature reviews," in *Proc. IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, Oct. 2020, pp. 1099–1100, doi: <https://doi.org/10.1109/COMPSAC48688.2020.00121>.

- [17] V. Garousi, M. Felderer, and M. V. Mäntylä, "Guidelines for including grey literature and conducting multivocal literature reviews in software engineering," *Information and Software Technology*, vol. 106, pp. 101–121, 2019, doi: <https://doi.org/10.1016/j.infsof.2018.09.006>.
- [18] T. Dykstra, "ASP.NET Core security topics," *Microsoft Learn*, Nov. 2024. [Online]. Available: <https://learn.microsoft.com/en-us/aspnet/core/security/?view=aspnetcore-10.0>
- [19] J. Popay *et al.*, *Guidance on the Conduct of Narrative Synthesis in Systematic Reviews*, Version 1.0. ESRC Methods Programme, 2006. [Online]. doi: <https://doi.org/10.13140/2.1.1018.4643>.
- [20] G. Roshan, "Architecting secure web applications: Best practices for software architects," *Medium*, Nov. 2024. [Online]. Available: <https://medium.com/@gavandiroshan/architecting-secure-web-applications-best-practices-for-software-architects-7639f2445bea>
- [21] J. Kanjilal, "How to improve API security in ASP.NET Core," *InfoWorld*, Aug. 2023. [Online]. Available: <https://www.infoworld.com/article/2334732/how-to-improve-api-security-in-asp-net-core.html>
- [22] L. Rodríguez, "Top 20 ASP.NET security interview questions (+ answers)," *Medium*, Jun. 2023. [Online]. Available: <https://medium.com/bytewidth/top-20-asp-net-security-interview-questions-answers-b780ea071a56>
- [23] T. Kalos, "ASP.NET security best practices," *Escape Blog*, Aug. 2024. [Online]. Available: <https://escape.tech/blog/asp-dot-net-security/>
- [24] Ramotion, "What is cross-origin resource sharing (CORS)?," Oct. 2023. [Online]. Available: <https://www.ramotion.com/blog/what-is-cors-in-web-development/>
- [25] R. Anderson, "Introduction to authorization in ASP.NET Core," *Microsoft Learn*, Dec. 2024. [Online]. Available: <https://learn.microsoft.com/en-us/aspnet/core/security/authorization/introduction?view=aspnetcore-10.0>
- [26] P. Jason, "ASP.NET website hosting: Best practices for securing your web applications," *AccuWeb Hosting Blog*, Dec. 2024. [Online]. Available: <https://www.accuwebhosting.com/blog/asp-net-website-hosting-best-practices-for-secure-your-web-applications/>
- [27] ByteHide, "ASP.NET security interview questions and answers (2023)," May 2023. [Online]. Available: <https://www.bytehide.com/blog/asp-net-security-interview-questions>

- [28] K. Kara, "Authentication and authorization in ASP.NET Core: A comprehensive guide," *Medium*, Dec. 2023. [Online]. Available: <https://medium.com/@kerimkkara/authentication-and-authorization-in-asp-net-core-a-comprehensive-guide-2025-84ff96df7264>
- [29] R. Patel, "A beginner's guide to ASP.NET Core Identity for authentication and authorization (.NET 8)," *Medium*, Nov. 2024. [Online]. Available: <https://medium.com/@ravipatel.it/a-beginners-guide-to-asp-net-core-identity-for-authentication-and-authorization-with-net-8-e6c-8deb612f4>
- [30] R. Anderson and K. Larkin, "Safe storage of app secrets in development in ASP.NET Core," *Microsoft Learn*, Nov. 2024. [Online]. Available: <https://learn.microsoft.com/en-us/aspnet/core/security/app-secrets?view=aspnetcore-10.0&tabs=windows>
- [31] F. Hasan and R. Anderson, "Prevent cross-site request forgery (XSRF/CSRF) attacks in ASP.NET Core," *Microsoft Learn*, Oct. 2024. [Online]. Available: <https://learn.microsoft.com/en-us/aspnet/core/security/anti-request-forgery?view=aspnetcore-10.0>
- [32] A. Kumar, "Avoiding cross-site scripting (XSS) attacks in C# and .NET Core," *C# Corner*, Apr. 2024. [Online]. Available: <https://www.c-sharpcorner.com/article/avoiding-cross-site-scripting-xss-attacks-in-c-sharp-and-net-core/>
- [33] S. Kataria, "Securing your financial data with ASP.NET Core solutions," *QServices*, Nov. 2024. [Online]. Available: <https://www.qservicesit.com/securing-your-financial-data-with-asp-net-core-solutions>
- [34] D. Rathod, "Complete guide to content security policy (CSP) in ASP.NET," *Atharva System*, Nov. 2024. [Online]. Available: <https://www.atharvasystem.com/the-complete-guide-to-content-security-policy-csp-in-asp-net/>
- [35] K. E. (ND), "10 best practices to secure ASP.NET Core MVC web applications," *Syncfusion Blog*, Dec. 2024. [Online]. Available: <https://www.syncfusion.com/blogs/post/10-practices-secure-asp-net-core-mvc-app>
- [36] A. Mohanty, "Web API security in .NET Core," *C# Corner*, Oct. 2023. [Online]. Available: <https://www.c-sharpcorner.com/article/web-api-security-in-net-core/>
- [37] Sanjay, "Broken access control in ASP.NET Core: OWASP Top 10," *Pro Code Guide*, Oct. 2023. [Online]. Available: <https://procodeguide.com/programming/broken-access-control-in-aspnet-core/>

- [38] S. X. (ND), "Creating fast and secure web API projects in .NET," *Softacom*, Jun. 2024. [Online]. Available: <https://www.softacom.com/blog/development/creating-fast-and-secure-web-api-project-in-net/>
- [39] D. M. Edirisinghe, "JWT authentication: Basics and best practices," *Medium*, Feb. 2023. [Online]. Available: <https://medium.com/@darshana-edirisinghe/jwt-security-concerns-f79e63ff4871>
- [40] P. Mehta, "Authentication and authorization in .NET Core," *Positiwise*, Jul. 2024. [Online]. Available: <https://positiwise.com/blog/authentication-and-authorization-in-net-core>
- [41] Riza (ND), "Best practices to secure your .NET Core application," *Codementor*, Oct. 2024. [Online]. Available: <https://www.codementor.io/@riza/best-practices-to-secure-your-net-core-application-2l0aue30mj>
- [42] J. Panchal, "Secure your APIs with .NET Core: Authentication and authorization," *Innovura Technologies*, May 2024. [Online]. Available: <https://innovuratech.com/implementing-authentication-and-authorization-in-net-core-apis/>
- [43] I. Baciú, "Building secure .NET applications," *Infobest Software Outsourcing Company*, Aug. 2024. [Online]. Available: <https://www.infobest.ro/building-secure-net-applications/>
- [44] S. Getachew, "Implementing role-based authorization in ASP.NET Core 8 Web API: Best practices for secure and scalable APIs," *Medium*, Nov. 2024. [Online]. Available: <https://medium.com/@solomongetachew112/implementing-role-based-authorization-in-asp-netc259f919fb5d>
- [45] M. Baldwin and T. Lanfear, "Best practices for protecting secrets," *Microsoft Learn*, Aug. 2024. [Online]. Available: <https://learn.microsoft.com/en-us/azure/security/fundamentals/secrets-best-practices>
- [46] Programmers.io Team, "Best practices for building secure web applications with ASP.NET Core," *Programmers.io*, Dec. 2024. [Online]. Available: <https://programmers.io/blog/building-web-application-with-asp-net-core/>
- [47] Peruzzi Services Limited, "How to deal with sensitive data in .NET," May 2024. [Online]. Available: <https://www.linkedin.com/pulse/how-deal-sensitive-data-net-peruzziservices-sqwtf/>
- [48] N. S. Filho, "Implementation and challenges of CORS in web applications developed with C#: A technical and practical analysis," *Leaders Tec*, 2024. [Online]. Available: <https://leaders.tec.br/article/26e6c8>

- [49] D. Galvan and R. Anderson, "Enforce HTTPS in ASP.NET Core," *Microsoft Learn*, Oct. 2024. [Online]. Available: <https://learn.microsoft.com/en-us/aspnet/core/security/enforcing-ssl?view=aspnetcore-10.0&tabs=visual-studio%2Clinux-sles>
- [50] R. Anderson, "Prevent cross-site scripting (XSS) in ASP.NET Core," *Microsoft Learn*, Sep. 2024. [Online]. Available: <https://learn.microsoft.com/en-us/aspnet/core/security/cross-site-scripting?view=aspnetcore-10.0>
- [51] A. Aborujilah, J. Adamu, S. M. Shariff, and Z. A. Long, "Descriptive analysis of built-in security features in web development frameworks," in *Proc. 16th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, 2022, pp. 1–8, doi: <https://doi.org/10.1109/IMCOM53663.2022.9721750>.