Cadena 1.

FS.1						
Datos generales						
Título	ASP.NET Core security topics					
Autores	Microsoft					
Fecha	11/11/2024					
Fuente	Microsoft.com					
Tipo de publicación (libro, revista, tesis, etc.)	Documentación oficial					
Referencia o detalles de la publicación	https://learn.microsoft.com/en-us/aspnet/core/security/?view=aspnetcore-9.0					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	X	No disponible	No disponible	x	x	No disponible

Datos para la síntesis			
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	Authentication	Authorization
	Descripción mecanismo	Authentication is a process in which a user provides credentials that are then compared to those stored in an operating system, database, app or resource.	users authenticate successfully, and can then perform actions that they're authorized for, during an authorization process. The authorization refers to the process that determines what a user is allowed to do.
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	No disponible	No disponible
	Descripción	No disponible	No disponible
PI-3	Nombre de la amenaza asociada al mecanismo	No disponible	No disponible

Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Descripción amenaza	No disponible	No disponible
PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo Descripción de la buena práctica	Managed identities are a secure way to authenticate to services without needing to store credentials in code, environment variables, or configuration files. Never store passwords or other sensitive data in configuration provider code or in plain text configuration files. The Secret Manager tool can be used to store secrets in development. Don't use production secrets in development or test environments. Specify secrets outside of the project so that they can't be accidentally committed to a source code repository.	No disponible
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Descripción de la mala práctica	Resource Owner Password Credentials Grant Exposes the user's password to the client. Is a significant security risk. Should only be used when other authentication flows are not possible.	No disponible
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Reto	No disponible	No disponible
	Descripción	No disponible	No disponible

FS.2	
Datos generales	
Título	Safe storage of app secrets in development in ASP.NET Core
Autores	Microsoft, Rick Anderson y Kirk Larkin
Fecha	11/04/2024

Fuente	Microsoft.com					
Tipo de publicación (libro, revista, tesis, etc.)	Documentación oficial					
Referencia o detalles de la publicación	https://learn.microsoft.com/en-us/aspnet/core/security/app-secrets?view=aspnetcore-9.0&tabs=windows					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	No disponible	X	No disponible	X	X	No disponible

Datos para la síntesis		
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	Authentication
	Descripción mecanismo	No disponible
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	No disponible
	Descripción	a database connection string stored in appsettings.json should not include a password. Instead, store the password as a secret, and include the password in the connection string at runtime. For example: dotnet user-secrets set "DbPassword" "`<secret value>`"
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	No disponible
	Descripción amenaza	No disponible
PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo	Secret Manager
	Descripción de la buena práctica	The Secret Manager tool stores sensitive data during application development. In this context, a piece of sensitive data is an app secret. App secrets are stored in a separate location from the project tree. The app secrets are associated with a specific project or shared across several projects. The app secrets aren't checked into source control The Secret Manager tool hides implementation details, such as where and how the values are stored. You can use the tool without knowing these implementation details. The values are stored in a JSON file in the local machine's user profile folder

		No disponible
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo	No disponible
	Descripción de la mala práctica	Never store passwords or other sensitive data in source code or configuration files. Production secrets shouldn't be used for development or test. Secrets shouldn't be deployed with the app. Don't write code that depends on the location or format of data saved with the Secret Manager tool. These implementation details may change. For example, the secret values aren't encrypted.
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Reto	No disponible
	Descripción	No disponible

FS.3						
Datos generales						
Título	Authentication and Authorization in ASP.NET Core: A Comprehensive Guide					
Autores	Kerim Kara					
Fecha	16/12/2023					
Fuente	Medium.com					
Tipo de publicación (libro, revista, tesis, etc.)	Blog					
Referencia o detalles de la publicación	https://medium.com/@kerimkkara/authentication-and-authorization-in-asp-net-core-a-comprehensive-guide-dfb8fb806ac7					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	X	X	X	X	X	No disponible

Datos para la síntesis			
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	Authorization	Authentication
	Descripción mecanismo	is the process of determining what actions a user is allowed to perform within an application. It involves checking whether the authenticated user has the necessary permissions or belongs to the appropriate role to access a particular resource or perform a specific operation. Authorization is crucial for enforcing access control and protecting sensitive data or functionality. The [Authorize] attribute is a powerful tool for enforcing access control in ASP.NET Core. When applied to a controller or action, it triggers the framework's authorization process, which verifies whether the current user is authenticated and has the necessary permissions to access the requested resource.	is the process of verifying the identity of a user, ensuring they are who they claim to be. This is typically done by presenting credentials, such as a username and password, and validating them against a trusted source, such as a database or an external authentication provider. Once authenticated, the user is assigned an identity, which is then used for subsequent authorization checks. Cookie authentication is one of the most common and straightforward authentication schemes in ASP.NET Core. It relies on HTTP cookies to store the user's authentication information, such as a session ID or an encrypted token. The cookie authentication middleware handles the process of

		By default, the [Authorize] attribute allows access to authenticated users regardless of their roles or claims. However, you can customize its behavior by specifying additional parameters or combining it with other authorization attributes Authorization policies allow you to define fine-grained rules for determining whether a user is authorized to perform a specific action.	creating and validating cookies, allowing users to remain authenticated across multiple requests. Claim-based authentication is a flexible and powerful authentication mechanism that revolves around the concept of claims. A claim represents a piece of information about the user, such as their name, email address, or role. By using claims, you can easily extend the user's identity with additional data and make authorization decisions based on these claims. ASP.NET Core provides built-in support for securing APIs using JSON Web Tokens (JWT) and the JWT bearer authentication scheme. JWTs are self-contained tokens that contain information about the user and their permissions. By validating the integrity and authenticity of a JWT, you can trust the claims it contains and authenticate API requests.
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	Financial Appliacations	
	Descripción	Authorization in financial applications is typically based on user roles and permissions. Different roles, such as customers, employees, or administrators, have varying levels of access to financial data and functionalities. Role-based authorization ensures that users can only perform actions or access resources that are appropriate for their role and level of authorization.	In e-commerce applications, authentication and authorization are essential to protect customer data, secure payment transactions, and enforce access control for administrative functions Authentication mechanisms, such as username/password authentication or social media sign-in, allow customers to create accounts, log in, and access personalized features like order history or saved addresses.
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	Cross-Site Scripting (XSS) Cross-Site Request Forgery (CSRF)	Brute-Force Attacks Session Management Password Storage Cross-Site Scripting (XSS)
	Descripción amenaza		
PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo		Two-factor authentication (2FA)
	Descripción de la buena práctica	Implement CSRF protection mechanisms, such as anti-forgery tokens, to prevent attackers from executing unauthorized actions on behalf of authenticated users. Protect against XSS attacks by properly encoding user input and validating data before rendering it in HTML or JavaScript.	adds an extra layer of security to the authentication process by requiring users to provide additional verification, typically in the form of a one-time password or a biometric factor. Implementing 2FA can significantly reduce the risk of

		implementing error logging and monitoring to track authentication and authorization failures.	unauthorized access, especially for sensitive applications or those handling confidential information. Implement account lockout policies and rate limiting to protect against brute-force attacks that attempt to guess user credentials. Use secure session management techniques, such as session timeouts, secure cookie attributes, and session regeneration, to prevent session hijacking or session fixation attacks. Store passwords securely by using strong hashing algorithms, salting, and iteration counts to protect against password cracking attempts. Protect against XSS attacks by properly encoding user input and validating data before rendering it in HTML or JavaScript. implementing error logging and monitoring to track authentication and authorization failures. For Razor Pages: Applying the [Authorize] attribute at the page or handler level to restrict access to authenticated users. Using claims or roles to make fine-grained authorization decisions within handlers. Displaying personalized content based on the user's authentication status or roles using the AuthorizeView tag helper. Redirecting unauthenticated users to the login page using the [Authorize] attribute or custom authorization filters. Protecting sensitive pages or actions with additional authorization checks, such as checking for specific claims or custom requirements.
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo		
	Descripción de la mala práctica	exposing sensitive information in error messages	exposing sensitive information in error messages
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Reto	No disponible	No disponible
	Descripción	No disponible	No disponible

FS.4						
Datos generales						
Título	Implementing Role-Based Authorization in ASP.NET Core 8 Web API: Best Practices for Secure and Scalable APIs					
Autores	Samuel Getachew					
Fecha	10/07/2024					
Fuente	Medium.com					
Tipo de publicación (libro, revista, tesis, etc.)	Blog					
Referencia o detalles de la publicación	https://medium.com/@solomongetachew112/implementing-role-based-authorization-in-asp-net-c259f919fb5d					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	No disponible	X	No disponible	X	No disponible	No disponible

Datos para la síntesis		
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	Authorization
	Descripción mecanismo	No disponible
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	No disponible
	Descripción	Admin panels where only administrators have full access. Subscription-based services where different levels of access are granted based on the user's plan. Multitenant applications where users from different organizations have distinct permissions.
PI-3	Nombre de la amenaza asociada al mecanismo	No disponible

Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Descripción amenaza	No disponible
PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo	
	Descripción de la buena práctica	-Use Policies Over Roles Directly: While it's tempting to use the RequireRole method directly, policies offer more flexibility. For example, you can create policies that require multiple roles or other conditions, making your authorization logic more maintainable. -Secure API Endpoints with Both Authentication and Authorization: Always combine role-based authorization with proper authentication mechanisms like JWT tokens to ensure the API is secure from unauthorized access. -Keep Sensitive Endpoints Separate: Avoid mixing sensitive administrative actions with general user actions. Isolate admin and user controllers for better security management. -Implement Logging and Auditing: For security-sensitive applications, it's essential to log access to critical endpoints and review these logs regularly to ensure there are no unauthorized access attempts. -Regularly Update Roles and Permissions: As your application grows, user roles may need to be updated. Ensure that you have a process in place for modifying roles and permissions.
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo	No disponible
	Descripción de la mala práctica	No disponible
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Reto	No disponible
	Descripción	No disponible

FS.5	
Datos generales	
Título	Architecting Secure Web Applications: Best Practices for Software Architects

Autores	Roshan Gavandi					
Fecha	10/05/2024					
Fuente	Medium.com					
Tipo de publicación (libro, revista, tesis, etc.)	Bllog					
Referencia o detalles de la publicación	https://roshancloudarchitect.me/architecting-secure-web-applications-best-practices-for-software-architects-7639f2445bea					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	X	No disponible	X	X	No disponible	No disponible

Datos para la síntesis

PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	Data Protection API	Authentication	anti-CSRF token
	Descripción mecanismo	simplifies encryption for protecting sensitive data, such as session tokens, cookies, and user information. This API manages encryption keys securely and automatically handles key rotation.		
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	No disponible	No disponible	No disponible
	Descripción	No disponible	No disponible	No disponible
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo			Cross-Site Request Forgery (CSRF)
	Descripción amenaza			CSRF exploits the trust that a website has in a user's browser. It tricks authenticated users into making unwanted requests, such as changing passwords or making purchases.
PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo			
	Descripción de la buena práctica	Use the Data Protection API to manage keys securely. Encrypt sensitive data, both at rest and in transit.	For JWT:	Use anti-CSRF tokens for all state-changing requests.

		Regularly rotate encryption keys to mitigate security risks.	Use Strong Signing Algorithms: Implement RS256 with private/public key encryption to secure JWT tokens. Short Token Expiration: Set token expiration times to minimize the risk of long-term token theft. Use HTTPS: Ensure all communication between Angular and .NET Core is encrypted over HTTPS to prevent man-in-the-middle attacks. Enforce strong password policies Use HTTP-only cookies to store session tokens, ensuring they cannot be accessed via JavaScript Implement short-lived session tokens with secure expiration policies to minimize session hijacking.	Validate tokens server-side to ensure authenticity.
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo	No disponible	No disponible	No disponible
	Descripción de la mala práctica	No disponible	No disponible	No disponible
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Reto	No disponible	No disponible	No disponible
	Descripción	No disponible	No disponible	No disponible

FS.6

Datos generales

Título	Prevent Cross-Site Request Forgery (XSRF/CSRF) attacks in ASP.NET Core
Autores	Microsoft, Fiyaz Hasan y Rick Anderson
Fecha	10/21/2024
Fuente	Microsoft.com
Tipo de publicación (libro, revista, tesis, etc.)	Doocumentación oficial
Referencia o detalles de la publicación	https://learn.microsoft.com/en-us/aspnet/core/security/anti-request-forgery?view=aspnetcore-8.0

Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	No disponible	X	X	X	No disponible	No disponible

Datos para la síntesis		
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	anti-CSRF token
	Descripción mecanismo	.
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	
	Descripción	An example of a CSRF attack: A user signs into www.good-banking-site.example.com using forms authentication. The server authenticates the user and issues a response that includes an authentication cookie. The site is vulnerable to attack because it trusts any request that it receives with a valid authentication cookie. The user visits a malicious site, www.bad-crook-site.example.com. The malicious site, www.bad-crook-site.example.com, contains an HTML form similar to the following example: <h1>Congratulations! You're a Winner!</h1> <form action="https://good-banking-site.com/api/account" method="post"> <input type="hidden" name="Transaction" value="withdraw" /> <input type="hidden" name="Amount" value="1000000" /> <input type="submit" value="Click to collect your prize!" /> </form> The user selects the submit button. The browser makes the request and automatically includes the authentication cookie for the requested domain, www.good-banking-site.example.com. The request runs on the www.good-banking-site.example.com server with the user's authentication context and can perform any action that an authenticated user is allowed to perform. In addition to the scenario where the user selects the button to submit the form, the malicious site could: Run a script that automatically submits the form. Send the form submission as an AJAX request. Hide the form using CSS. These alternative scenarios don't require any action or input from the user other than initially visiting the malicious site.

PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	Cross-site request forgery
	Descripción amenaza	is an attack against web-hosted apps whereby a malicious web app can influence the interaction between a client browser and a web app that trusts that browser. These attacks are possible because web browsers send some types of authentication tokens automatically with every request to a website. This form of exploit is also known as a *one-click attack* or *session riding* because the attack takes advantage of the user's previously authenticated session.
PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo	
	Descripción de la buena práctica	Explicitly add an antiforgery token to a <form> element without using Tag Helpers with the HTML helper @Html.AntiForgeryToken We recommend use of AutoValidateAntiforgeryToken broadly for non-API scenarios. This attribute ensures POST actions are protected by default. The alternative is to ignore antiforgery tokens by default, unless ValidateAntiForgeryToken is applied to individual action methods. It's more likely in this scenario for a POST action method to be left unprotected by mistake, leaving the app vulnerable to CSRF attacks. All POSTs should send the antiforgery token. Tokens should be refreshed after the user is authenticated by redirecting the user to a view or Razor Pages page.
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo	No disponible
	Descripción de la mala práctica	No disponible
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Reto	No disponible
	Descripción	No disponible

FS.7	
Datos generales	
Título	Best practices for protecting secrets

Autores	Microsoft, M. Baldwin y Terry Lanfear					
Fecha	08/30/2024					
Fuente	Microsoft.com					
Tipo de publicación (libro, revista, tesis, etc.)	Documentación oficial					
Referencia o detalles de la publicación	https://learn.microsoft.com/en-us/azure/security/fundamentals/secrets-best-practices					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	No disponible	No disponible	X	X	X	No disponible

Datos para la síntesis		
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	**Secrets Management**
	Descripción mecanismo	. No disponible
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	No disponible
	Descripción	No disponible
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	unauthorized access
	Descripción amenaza	
PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo	
	Descripción de la buena práctica	Conduct an audit to identify secrets Before you can secure your secrets, you need to know where they are. Conducting a thorough audit of your systems and applications helps identify all the sensitive information that needs protection. Avoid hardcoding secrets use environment variables or configuration management tools that keep secrets out of your source code. This practice minimizes the risk of accidental exposure and simplifies the process of updating secrets.

		Use secure key stores Leveraging secure key stores ensures that your secrets are stored in a secure, encrypted location. Services like Azure Key Vault and Azure Managed HSM provide robust security features, including access control, logging, and automatic rotation. Implement secret scanning tools Regularly scanning your codebase for embedded secrets can prevent accidental exposure. Tools like Azure DevOps Credential Scanner and GitHub secret scanning feature can automatically detect and alert you to any secrets found in your repositories. Leverage managed identities Managed identities in Azure provide a secure way for applications to authenticate to Azure services without storing credentials in the code. By enabling managed identities for Azure resources, you can securely access Azure Key Vault and other services, reducing the need to handle secrets manually. Apply granular access control Follow the principle of least privilege by applying granular access control to your secrets. Use Azure role-based access control (RBAC) to ensure that only authorized entities have access to specific secrets. Rotate secrets regularly Secrets are susceptible to leakage or exposure over time. Regularly rotating your secrets reduces the risk of unauthorized access. Monitor and log access Enable logging and monitoring for your secret management system to track access and usage. Use Key Vault logging and/or services like Azure Monitor and Azure Event Grid, to monitor all activities related to your secrets. Implement network isolation Reduce the exposure of your secrets by implementing network isolation. Configure firewalls and network security groups to restrict access to your key vaults. Encrypt secrets at rest and in transit Ensure that your secrets are encrypted both at rest and in transit. Azure Key Vault securely stores secrets using envelope encryption, where Data Encryption Keys (DEKs) are encrypted by Key Encryption Keys (KEKs), providing an additional layer of security. Safe distribution of secrets When distributing secrets, ensure they are shared securely within and outside the organization. Use tools designed for secure sharing and include secret recovery procedures in your disaster recovery plans.
PI-5	Mala práctica asociada al mecanismo	

Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Descripción de la mala práctica	Embedding secrets directly into your code or configuration files is a significant security risk. If your codebase is compromised, so are your secrets.
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Reto	No disponible
	Descripción	No disponible

FS.8						
Datos generales						
Título	JWT authentication: Basics and best practices					
Autores	Darshana Mihiran Edirisinghe					
Fecha	02/18/2023					
Fuente	Medium.com					
Tipo de publicación (libro, revista, tesis, etc.)	Blog					
Referencia o detalles de la publicación	https://medium.com/@darshana-edirisinghe/jwt-security-concerns-f79e63ff4871					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	X	No disponible	X	X	No disponible	No disponible

Datos para la síntesis		
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	Authentication
	Descripción mecanismo	JWT JWTs are often used to authenticate users after they've provided valid credentials. When a user logs in, the server generates a JWT with information about the user (such as their user ID or email address) and signs it using a secret key. The server

		then sends the JWT back to the client, which includes it in all subsequent requests. The server can verify the JWT to ensure that the user is who they claim to be.
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	No disponible
	Descripción	No disponible
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	eavesdropping or man-in-the-middle (MITM) attacks
	Descripción amenaza	
PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo	
	Descripción de la buena práctica	When using JWTs, make sure to use strong cryptographic algorithms such as HMACSHA256, RSA with at least 2048-bit keys or better, and AES with a key length of 256 bits. Setting token expiry time is important for security. Tokens should not be valid indefinitely. Implementing an appropriate expiry time can protect the application from malicious users holding onto tokens indefinitely. There should be an option to revoke tokens if a user is no longer authorized to access the application. This can be achieved by maintaining a blacklist of revoked tokens. always validate the token's signature and integrity before processing the claims. Secure transmission of the token is critical to protect against interception,. It is recommended to encrypt JWTs if sensitive information is being transmitted. This will help to protect against malicious users tampering with the token or retrieving sensitive data from the token.
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo	No disponible
	Descripción de la mala práctica	No disponible
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Reto	No disponible
	Descripción	No disponible

FS.9						
Datos generales						
Título	How to deal with sensitive data in .NET					
Autores	Peruzzi Solutions Limited					
Fecha	23/05/2024					
Fuente	Linkedin.com					
Tipo de publicación (libro, revista, tesis, etc.)	Blog					
Referencia o detalles de la publicación	https://www.linkedin.com/pulse/how-deal-sensitive-data-net-peruzziservices-sqwtf/					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	X	No disponible	No disponible	No disponible	X	X

Datos para la síntesis		
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	Secrets Management
	Descripción mecanismo	provide a mechanism for storing sensitive data outside of the project's source code, preventing the data from being accidentally shared or checked into source control. This technique is particularly useful during development, as it allows developers to keep sensitive information such as API keys, connection strings, and other secrets out of their codebase and configuration files like appsettings.json.
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	No disponible
	Descripción	No disponible
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	No disponible
	Descripción amenaza	No disponible
PI-4	Buena práctica asociada al mecanismo	No disponible

Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Descripción de la buena práctica	No disponible
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo	
	Descripción de la mala práctica	hardcoding sensitive information, such as passwords, API keys, and connection strings, directly into the source code. storing sensitive data in appsettings.json without proper encryption or access controls still poses security risks, especially if the file is included in source control. So this technique is especially risky in open-source projects or when repositories are not properly secured. Developers might also overlook the need for secure communication channels, transmitting sensitive data over unencrypted connections. Using the same credentials or keys across different environments (development, testing, and production) increases the risk of exposure and misuse. the failure to implement proper access controls and audit trails for sensitive data access can obscure the detection of unauthorized access or data breaches, complicating security monitoring and response efforts.
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Reto	Editing User Secrets in Visual Studio
	Descripción	Visual Studio provides a more integrated and user-friendly approach to manage User Secrets, but it requires additional setup and might have compatibility issues with older or unsupported .NET Core versions.

FS.10	
Datos generales	
Título	TOP 20 ASP.NET Security Interview Questions (+Answers)
Autores	Lidia Rodríguez
Fecha	13/06/2023
Fuente	Medium.com
Tipo de publicación (libro, revista, tesis, etc.)	Bllog
Referencia o detalles de la publicación	https://medium.com/bytehide/top-20-asp-net-security-interview-questions-answers-b780ea071a56

Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	X	No disponible	X	X	No disponible	No disponible

Datos para la síntesis				
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	Anti-Forgery Tokens	Output encoding	CORS
	Descripción mecanismo	These tokens involve: Generating a unique token for each user session. Storing this token on the server and embedding it as a hidden field in HTML forms. Encrypting and validating the token when the form is submitted back to the server. If the token is missing or doesn't match the stored value, the server will reject the request as malicious. By validating the anti-forgery token in each HTTP POST request, ASP.NET ensures that the request originates from the original application and not from third-party sites.	Encode potentially unsafe characters in user-generated content, such as HTML and JavaScript, before rendering it on the page. Use HTML encoding functions like Html.Encode() or HttpUtility.HtmlEncode().	is a security feature that enables web applications to control which external domains are allowed access to their resources. By default, web browsers enforce a same-origin policy that restricts resources from being accessed by scripts or APIs originating from different domains. However, for legitimate use cases like fetching data from a public API or loading resources from a content delivery network (CDN), CORS provides a mechanism to relax the cross-origin restrictions.
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	No disponible	No disponible	No disponible
	Descripción	No disponible	No disponible	No disponible
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	Cross-Site Request Forgery (CSRF)	Cross-Site Scripting (XSS)	unauthorized access to sensitive data
	Descripción amenaza	is a security attack where an attacker tricks a user into executing unintended actions on a web application, while they're authenticated. For instance, an attacker might forge a request to transfer funds from the victim's account to the attacker's account.	is a security attack where an attacker injects malicious scripts (typically JavaScript) into trusted websites. The main aim is to steal sensitive information (like session cookies or personal data) from users who visit the compromised website.	
PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo			
	Descripción de la buena práctica		Input validation: Validate all user inputs, especially those that will be rendered as HTML.	

			Validate data length, type, format, and range to minimize the risk of accepting malicious input. Content Security Policy (CSP): Create a policy that restricts the browser from executing scripts that don't adhere to the policy. With CSP, define the allowed sources of scripts and other resources, making it difficult for attackers to inject malicious content. Request validation: Activate the built-in request validation feature in ASP.NET that blocks any requests containing potentially unsafe content by default	
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo	No disponible	No disponible	No disponible
	Descripción de la mala práctica	No disponible	No disponible	No disponible
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Reto	No disponible	No disponible	No disponible
	Descripción	No disponible	No disponible	No disponible

Cadena 2. Autenticación

FS.11	
Datos generales	
Título	Overview of ASP.NET Core authentication
Autores	Microsoft – Mike Rousos
Fecha	14/02/2024
Fuente	Microsoft.com
Tipo de publicación (libro, revista, tesis, etc.)	Documentación oficial
Referencia o detalles de la publicación	https://learn.microsoft.com/en-us/aspnet/core/security/authentication/?view=aspnetcore-9.0

Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	X	X	No disponible	No disponible	No disponible	No disponible

Datos para la síntesis			
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	Authentication	Authorization
	Descripción mecanismo	Authentication is the process of determining a user's identity.	Authorization is the process of determining whether a user has access to a resource.
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario		
	Descripción	• Authenticating a user. • Responding when an unauthenticated user tries to access a restricted resource.	
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	No disponible	
	Descripción amenaza		
PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo	No disponible	
	Descripción de la buena práctica		
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo	No disponible	
	Descripción de la mala práctica		
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Descripción	No disponible	

FS.12						
Datos generales						
Título	Web API Security in .NET Core					
Autores	C# Corner– Amit Mohanty					
Fecha	03/10/2023					
Fuente	www.c-sharpcorner.com					
Tipo de publicación (libro, revista, tesis, etc.)	Blog					
Referencia o detalles de la publicación	https://www.c-sharpcorner.com/article/web-api-security-in-net-core/					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	X	No disponible	X	No disponible	No disponible	No disponible

Datos para la síntesis					
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	Authentication and Authorization		HTTPS and Transport Security	CORS (Cross-Origin Resource Sharing)
	Descripción mecanismo	Authentication is the process of verifying the identity of a user or system, while authorization defines what actions a user or system is allowed to perform.		Enforcing HTTPS ensures that data transmitted between clients and your API is encrypted.	CORS policies determine which origins are allowed to access your API.
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	No disponible			
	Descripción				
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	Unauthorized Access	Data Breaches	Denial of Service (DoS) Attacks	Data Tampering
	Descripción amenaza	Malicious users may attempt to access sensitive data or perform actions they are not authorized for.	Unauthorized access to data can lead to data breaches, resulting in the exposure of confidential information.	Attackers can overwhelm an API by sending a large number of requests, causing it to become slow or unresponsive.	Data transmitted between the client and the API can be intercepted and modified.

PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo	No disponible			
	Descripción de la buena práctica				
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo	No disponible			
	Descripción de la mala práctica				
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Descripción	No disponible			

FS.13	
Datos generales	
Título	Authentication and Authorization in .NET Core
Autores	Positiwise – Parag Mehta
Fecha	24/07/2024
Fuente	positiwise.com

Tipo de publicación (libro, revista, tesis, etc.)	Blog					
Referencia o detalles de la publicación	https://positiwise.com/blog/authentication-and-authorization-in-net-core					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	X	No disponible	No disponible	X	No disponible	No disponible

Datos para la síntesis

PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	Authentication	Authorization
	Descripción mecanismo	Authentication in .NET Core refers to the process of determining the identity of a user… is a process where the identity of the users is verified by those who wish to attempt to access an application or a system.	Authorization… refers to the process of determining whether a user has access to a resource… is the process of determining the actions authenticated that users can perform within the application.
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	No disponible	
	Descripción		
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	No disponible	
	Descripción amenaza	No disponible	

PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo	Use ASP.NET Core Identity	Enable Multi-Factor Authorization (MFA)	Secure Sensitive Data With HTTPS	Use The Storage For Secrets	Update and Patch Dependencies	Monitor and Log Authentication Events	Prevent Injection Attack	Leverage OAuth2 and OpenID Connect For External Authentication
	Descripción de la buena práctica	You can use it for handling the authorization and authentication as it provides a strong framework to manage users, passwords, role-based access, and	It is crucial to add an extra layer of security enabling the MFA which helps verify the user identity via multiple ways like SMS, email, and	Use HTTPS to encrypt the data within the transit between the client and the server. This prevents the interception and the tampering of	Store your sensitive information like the API keys, connection strings, and other secrets.	Make sure that your .NET Core libraries and dependencies are up to date with the latest security patches.	Implement the logging and monitoring for the authentication and authorization events.	Validate and sanitize the user input to protect against SQL injection, cross-site scripting, and other injection attacks. You can use parameterized	To integrate external login providers such as Google, Facebook, or Microsoft, use OAuth2 and OpenID Connect. These protocols present secure

		claims-based authorization.	authenticator apps.	sensitive information including the authentication credentials.				queries and inbuilt validation frameworks to ensure data integrity.	methods for user authentication and authorization.
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo	No disponible							
	Descripción de la mala práctica								
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Descripción	No disponible							

FS.14						
Datos generales						
Título	Implementing Authentication and Authorization in .NET Core APIs					
Autores	Innovura – Jalpa Panchal					
Fecha	22/05/2024					
Fuente	innovuratech.com					
Tipo de publicación (libro, revista, tesis, etc.)	Blog					
Referencia o detalles de la publicación	https://innovuratech.com/implementing-authentication-and-authorization-in-net-core-apis/					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	X	No disponible	No disponible	X	No disponible	No disponible

Datos para la síntesis				
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	Authentication	Authorization	
	Descripción mecanismo	Authentication verifies the identity of users attempting to access a system or application, ensuring they are who they claim to be. It involves various methods such as passwords, biometrics, and multi-factor authentication.	authorization determines the actions and resources a user is permitted to access based on their authenticated identity. This involves defining roles, permissions, and access control policies.	
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	No disponible		
	Descripción			
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	No disponible		
	Descripción amenaza	No disponible		
PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo	Authorization	HTTPS	
	Descripción de la buena práctica	RBAC best practices such as role segregation, least privilege, and continuous access review.	HTTPS to encrypt data transmisión	input validation to prevent injection attacks, and strong password hashing algorithms to securely store user credentials… the importance of regular dependency updates to mitigate security vulnerabilities and the implementation of logging and monitoring mechanisms to detect and respond to security incidents proactively. Furthermore, we highlight the significance of employing design patterns to enhance the maintainability, scalability, and security
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo	No disponible		
	Descripción de la mala práctica			
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Descripción	No disponible		

FS.15

Datos generales

Título	How to improve API security in ASP.NET Core					
Autores	InfoWorld – Joydip Kanjilal					
Fecha	31/08/2023					
Fuente	infoworld.com					
Tipo de publicación (libro, revista, tesis, etc.)	Blog					
Referencia o detalles de la publicación	https://www.infoworld.com/article/2334732/how-to-improve-api-security-in-aspnet-core.html					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	X	No disponible	X	No disponible	No disponible	No disponible

Datos para la síntesis

PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	Authentication	Authorization		
	Descripción mecanismo	authentication is used to validate the identity of a user	authorization is used to grant or revoke access to specific resources in the application based on the user's access privileges.		
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	No disponible			
	Descripción				
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	No disponible			
	Descripción amenaza	No disponible			
PI-4	Buena práctica asociada al mecanismo	Authentication			CORS

Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Descripción de la buena práctica	Another technique used to authenticate users in an application is token-based authentication. Here a unique token is generated for each authenticated user, i.e., the token is generated once the identity of the user has been validated… You can also use API keys to authenticate users in an application. API keys are unique identifiers that are passed in the request header on each call to the API… you can take advantage of two-factor authentication	You should only store the data that you need and only for as long as you need it. Regularly clean up old or unnecessary data… secure stored passwords using hashing	implement the principle of least knowledge to provide only necessary access.	use CORS to thwart unauthorized access to your APIs from other domains.
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo	No disponible			
	Descripción de la mala práctica				
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Descripción	No disponible			

Cadena 3

FS.16	
Datos generales	
Título	Introduction to authorization in ASP.NET Core
Autores	Microsoft
Fecha	03/06/2022
Fuente	microsoft.com
Tipo de publicación (libro, revista, tesis, etc.)	Documentación official

Referencia o detalles de la publicación	https://learn.microsoft.com/en-us/aspnet/core/security/authorization/introduction?view=aspnetcore-8.0					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	X	X	No disponible	No disponible	No disponible	No disponible

Datos para la síntesis			
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	Authorization	Authentication
	Descripción mecanismo	Authorization refers to the process that determines what a user is able to do… Authorization is separate and distinct from authentication.	Authentication is the process of verifying a user's identity.
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario		
	Descripción	an administrative user is allowed to create a document library, add documents, edit documents, and delete them. A non-administrative user working with the library is only authorized to read the documents.	
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	No disponible	
	Descripción amenaza	No disponible	
PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo	No disponible	
	Descripción de la buena práctica	No disponible	
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo	No disponible	
	Descripción de la mala práctica		
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Descripción	No disponible	

FS.17						
Datos generales						
Título	A Beginner’s Guide to ASP.NET Core Identity for Authentication and Authorization (with .NET 8)					
Autores	Ravi Patel					
Fecha	04/10/2024					
Fuente	medium.com					
Tipo de publicación (libro, revista, tesis, etc.)	Blog					
Referencia o detalles de la publicación	https://medium.com/@ravipatel.it/a-beginners-guide-to-asp-net-core-identity-for-authentication-and-authorization-with-net-8-e6c8deb612f4					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	X	X	No disponible	No disponible	No disponible	No disponible

Datos para la síntesis					
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	Authentication		Authorization	
	Descripción mecanismo	Authentication is the process of identifying the user. It answers the question, “Who are you?”		Authorization comes after authentication and answers, “What are you allowed to do?”	
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	E-commerce websites	Content Management Systems (CMS)	Enterprise Applications	Social Media Platforms
	Descripción	Managing user accounts, tracking orders, and handling admin and customer roles.	Allowing only certain users (e.g., content editors) to modify or publish content.	Managing employee logins, tracking permissions, and enforcing security policies.	Handling user profiles, managing followers, and assigning moderator/admin roles for community management.
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	No disponible			
	Descripción amenaza	No disponible			

PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo	No disponible
	Descripción de la buena práctica	No disponible
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo	No disponible
	Descripción de la mala práctica	
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Descripción	No disponible

FS.18						
Datos generales						
Título	How to create a fast and secure WEB API project in .NET					
Autores	SOFTACOM – Serge X					
Fecha	05/06/2024					
Fuente	www.softacom.com					
Tipo de publicación (libro, revista, tesis, etc.)	Blog					
Referencia o detalles de la publicación	https://www.softacom.com/blog/development/creating-fast-and-secure-web-api-project-in-net/					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	X	No disponible	X	No disponible	No disponible	No disponible
Datos para la síntesis						

PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	Authentication	Authorization	CORS (Cross-Origin Resource Sharing)	HTTPS and Transport Security	Rate Limiting and IP Whitelisting
	Descripción mecanismo	Authentication is the process of verifying the identity of a user or system. JWT is a popular mechanism for securing web APIs by encoding information in a token that can be easily validated.	authorization defines what actions a user or system is allowed to perform.	CORS policies determine which origins are allowed to access your API.	Enforcing HTTPS ensures that data transmitted between clients and your API is encrypted.	It is also possible to implement rate limiting in order to prevent abuse of your API and consider IP whitelisting to restrict the number of requests a client can make within a specific timeframe.
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	No disponible				
	Descripción	No disponible				
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	Unauthorized Access	Data Breaches	Denial of Service (DoS) Attacks	Data Tampering	
	Descripción amenaza	Malicious users may attempt to access sensitive data or perform actions they are not authorized for.	Unauthorized access to data can lead to data breaches, resulting in the exposure of confidential information.	Attackers can overwhelm an API by sending a large number of requests, causing it to become slow or unresponsive.	Data transmitted between the client and the API can be intercepted and modified.	
PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo	Input validation				
	Descripción de la buena práctica	Preventing SQL injection and XSS attacks is crucial. Always validate and sanitize user input and apply parameterized queries while interacting with the database.				
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo	No disponible				
	Descripción de la mala práctica					
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Descripción	No disponible				

FS.19						
Datos generales						
Título	Broken Access Control in ASP.NET Core – OWASP Top 10					
Autores	procodeguide – Sanjay					
Fecha	29/08/2022					
Fuente	www. procodeguide.com					
Tipo de publicación (libro, revista, tesis, etc.)	Blog					
Referencia o detalles de la publicación	https://procodeguide.com/programming/broken-access-control-in-aspnet-core/					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	X	No disponible	X	X	No disponible	No disponible

Datos para la síntesis		
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	Authorization
	Descripción mecanismo	is defined as a set of policies or mechanisms to provide control over the resources of the application.
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	No disponible
	Descripción	No disponible

PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo									
	Descripción amenaza	Not implementing the principle of denial by default i.e. access to restricted information or actions should be by default not allowed for anyone and should be enabled on a need basis for a specific user, user role or group i.e. restricted information should not be available to all users by default and then you disable it for the users who are not allowed to access it.	Getting access to the restricted resource or information or performing restricted action on the data by bypassing the access control mechanism of the application by manipulating the URL of the applications i.e. URL tampering that involves modification of the URL parameters, modification of the HTML page, or by using an attack tool to modify the HTTP request being sent to the application.	Getting access to the details or data belonging to some other user by making use of the unique identifier of the data of the other user. i.e. attackers can modify the input unique identifier of the data to the request to access data belonging to other users.	By some means changing the privileges assigned to the user so that it is able to get the higher privileges in the application i.e. either acting as an authenticated & authorized logged-in user without performing a successful login or a normal standard user gaining the elevation to an admin user.	Exploiting the vulnerabilities in the security token used by the application i.e. replaying or tampering with the most popularly used JSON Web Token (JWT) access token or a cookie or hidden field to gain access to the restricted data or actions which are not allowed to the user.	Misconfiguration in CORS (Cross-Origin resource sharing) allows the untrusted or unauthorized origins (websites) to gain access to the application API and enable attackers to trick users into performing actions without their knowledge.	force browsing to the authenticated or restricted pages by unauthenticated & unauthorized users impersonating an authenticated & authorized user or standard users impersonating an admin user.	Misconfiguration or missing access control in security for HTTP methods PUT, POST & DELETE allows unauthorized users or attackers to gain access to the restricted application API by exploiting the missing access control configuration.
PI-4	Buena práctica asociada al mecanismo									

Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Descripción de la buena práctica	policies that prevent users from accessing resources or performing actions that are not permitted to the user i.e. users should not be able to view information or data which does not belong to them or is not authorized to access and also should not be able to perform non-permitted actions (like add, modify, delete, etc.) on the data.
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo	No disponible
	Descripción de la mala práctica	
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Descripción	No disponible

Cadena 4

FS.20						
Datos generales						
Título	Best Practices to Secure Your .NET Core Application					
Autores	Ravi Patel					
Fecha	15/10/2024					
Fuente	www.codementor.io					
Tipo de publicación (libro, revista, tesis, etc.)	Blog					
Referencia o detalles de la publicación	https://www.codementor.io/@riza/best-practices-to-secure-your-net-core-application-2l0aue30mj					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	No disponible	No disponible	X	X	No disponible	No disponible

Datos para la síntesis				
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	No disponible		
	Descripción mecanismo	No disponible		
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	No disponible		
	Descripción	No disponible		
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	Cross-Site Scripting (XSS)	Cross-Site Request Forgery (CSRF)	SQL Injection
	Descripción amenaza	Allows attackers to inject malicious scripts into web pages viewed by users.	Tricks a user into performing actions on a site where they are authenticated.	Occurs when attackers manipulate SQL queries to gain unauthorized access to the database.
PI-4	Buena práctica asociada al mecanismo			

Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Descripción de la buena práctica	HTTPS by Default: encourages the use of HTTPS to encrypt data transmitted between clients and servers. Enforce HTTPS and HSTS: Ensure that all communications between the client and server are secure by enforcing HTTPS in your application. Authentication and Authorization: Use ASP.NET Core Identity to implement secure authentication and authorization in your web app. Role-based authentication ensures that users only access permitted areas of the application…Use ASP.NET Core Identity for user authentication…For API-based authentication, use JWT (JSON Web Token) with bearer tokens…Apply [Authorize] attribute to secure endpoints or actions, and use role-based or policy-based authorization Prevent Cross-Site Scripting (XSS): To prevent XSS attacks, always validate and sanitize user input. Use built-in mechanisms such as Razor's automatic HTML encoding for output. SQL Injection Prevention: prevent SQL Injection by using parameterized queries with Entity Framework or Dapper, rather than directly concatenating user input into SQL statements. Protect Against CSRF Attacks: Enable CSRF protection in .NET Core by using anti-forgery tokens in forms and API requests. Razor pages automatically include CSRF tokens in forms. CSRF tokens are automatically generated for forms in Razor Pages and MVC applications. For AJAX requests, include the anti-forgery token in headers. Use Strong Password Policies: Configure ASP.NET Core Identity to enforce strong password rules Implement Data Protection API (DPAPI): Use the Data Protection API to encrypt sensitive data. By default, it is used for things like encrypting authentication cookies Secure Cookies: Mitigates Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF) risks…Set the HttpOnly, Secure, and SameSite attributes on cookies to prevent them from being accessed through JavaScript and to limit when they are sent. Enable Content Security Policy (CSP): Mitigates XSS by restricting the types of resources that can be loaded on your site…Set CSP headers to allow only trusted sources for scripts and styles. Limit Request Size: Prevents Denial of Service (DoS) attacks by limiting the size of the request body. Rate Limiting: Mitigates brute force and DoS attacks by limiting the number of requests from a specific IP. Log Sensitive Action: Helps to detect unusual activities and identify potential threats. Avoid Using Secrets in Code: Prevent sensitive information like API keys, connection strings, and passwords from being exposed in source code.

		Secure APIs: Prevents unauthorized access to API endpoints. Monitor and Update Dependencies: Vulnerable or outdated packages can expose your application to attacks. Security Headers: Adds protection against various types of attacks. How: Add security headers like Strict-Transport-Security, X-Content-Type-Options, and X-Frame-Options Advanced Security: OAuth, JWT, and Identity: Leverage OAuth 2.0 and OpenID Connect for secure third-party authentication (Google, Microsoft, etc.). When building APIs, secure them with JWT (JSON Web Tokens). Input Validation and Output Encoding: Prevents Cross-Site Scripting (XSS) and SQL Injection. Use model validation with data annotations
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo	No disponible
	Descripción de la mala práctica	
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Descripción	No disponible

FS.21	
Datos generales	
Título	Securing Your Financial Data with ASP.NET Core Solutions
Autores	Services Quality Redefined
Fecha	19/06/2024

Fuente	www.qservicesit.io					
Tipo de publicación (libro, revista, tesis, etc.)	Blog					
Referencia o detalles de la publicación	https://www.qservicesit.com/securing-your-financial-data-with-asp-net-core-solutions					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	X	X	No disponible	No disponible	No disponible	No disponible

Datos para la síntesis					
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	Authentication	Authorization	Data protection	HTTPS Enforcement
	Descripción mecanismo	Authentication verifies user identities by comparing credentials, such as usernames and passwords, with stored data.	Authorization determines the actions users are permitted to perform within the server, database, or application.	Data Protection API secures sensitive data through key management and rotation, ensuring data confidentiality and integrity both at rest and in transit.	Enforcing HTTPS is crucial for secure communication. ASP.NET Core allows easy configuration of HTTPS settings, protecting data during transmission.
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario				
	Descripción	Protecting financial data is crucial for organizations in the financial services sector, where keeping sensitive information confidential, intact, and accessible is very important. ASP.NET Core offers strong security features to enhance these efforts			
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	Cross-Site Scripting (XSS) attacks	SQL injection attacks	Cross-Site Request Forgery (XSRF/CSRF) attacks	Open redirect attacks
	Descripción amenaza				
PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo	No disponible			
	Descripción de la buena práctica	No disponible			
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo	No disponible			
	Descripción de la mala práctica				

PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Descripción	No disponible

FS.22						
Datos generales						
Título	Best Practices for Building Secure Web Applications with ASP.NET Core					
Autores	PIO TEAM					
Fecha	19/05/2023					
Fuente	www.programmers.io					
Tipo de publicación (libro, revista, tesis, etc.)	Blog					
Referencia o detalles de la publicación	https://programmers.io/blog/building-web-application-with-asp-net-core/					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	No disponible	No disponible	No disponible	X	No disponible	No disponible

Datos para la síntesis		
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	No disponible
	Descripción mecanismo	No disponible
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	No disponible
	Descripción	No disponible

PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	No disponible				
	Descripción amenaza	No disponible				
PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo		Secure Authentication and Authorization	Data protection	Secure Communication Protocols	Cross-Site Scripting (XSS) Prevention
	Descripción de la buena práctica	You can implement input validation and sanitization techniques to prevent cross-site scripting (XSS) attacks. Use parameterized queries and stored procedures to prevent SQL injection attacks, and avoid using deprecated or vulnerable components and libraries.	Implement strong authentication mechanisms, such as multi-factor authentication (MFA) and strong password policies. You should use secure protocols (e.g., OAuth, OpenID Connect) for authentication and authorization and employ role-based access control (RBAC) to ensure proper authorization and permissions management.	You need to encrypt sensitive data at rest using strong encryption algorithms, hash passwords, and salted hashes to protect user credentials. You can also employ secure data storage techniques, such as encryption and proper access control and implement data anonymization or pseudonymization where applicable.	Use HTTTPS with strong SSL/TLS protocols and cipher suites to encrypt data in transit and implement HSTS (HTTP Strict Transport Security) to enforce secure communication. You should regularly update SSL/TLS certificates and avoid using self-signed certificates.	Apply output encoding to user-generated content and dynamically generated HTML. You can use content security policies (CSP) to restrict the execution of malicious scripts and implement input validation and filtering to prevent XSS attacks.
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo	No disponible				
	Descripción de la mala práctica					
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Descripción	No disponible				

FS.23						
Datos generales						
Título	ASP.NET Security Interview Questions and Answers					
Autores	Bytehide					
Fecha	31/05/2023					
Fuente	www.bytehide.io					
Tipo de publicación (libro, revista, tesis, etc.)	Blog					
Referencia o detalles de la publicación	https://www.bytehide.com/blog/asp-net-security-interview-questions					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	X	X	X	X	No disponible	No disponible

Datos para la síntesis					
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	CORS		HTTPS	JWT
	Descripción mecanismo	Cross-Origin Resource Sharing (CORS) is a security feature that enables web applications to control which external domains are allowed access to their resources. By default, web browsers enforce a same-origin policy that restricts resources from being accessed by scripts or APIs originating from different domains. However, for legitimate	ASP.NET encryption and decryption features help protect sensitive data by transforming it into a non-readable format (encryption) and converting it back to a readable format when required (decryption). Some common examples of sensitive data include API keys, connection strings, and user personal information.	Using HTTPS and TLS in an ASP.NET application ensures secure communication by encrypting data transmitted between the server and client.	JSON Web Tokens (JWT) are a compact, URL-safe method of representing claims to be transferred between two parties. JWT is often used to implement authentication and authorization in web applications, as they can be easily created, transmitted, and consumed.

		use cases like fetching data from a public API or loading resources from a content delivery network (CDN), CORS provides a mechanism to relax the cross-origin restrictions.					
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	Authorization			Authentication		
		Authorization Role-Based Security	Authorization Claims-Based Security	Windows Authentication	Forms Authentication	OAuth, OpenID Connect, SAML	
	Descripción	Ideal for simple applications, with clearly defined user roles, and a limited number of permissions.	Ideal for complex applications, with multiple user types, numerous permissions, and granular access control requirements.	Ideal for intranet applications with Windows-based systems and Active Directory.	Can use custom/user-defined credential storage	Ideal for modern applications needing scalable, federated identity management.	
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	Cross-Site Request Forgery (CSRF)	Cross-Site Scripting (XSS)	Insecure Direct Object Reference (IDOR)			
	Descripción amenaza	is a security attack where an attacker tricks a user into executing unintended actions on a web application, while they're authenticated.	is a security attack where an attacker injects malicious scripts (typically JavaScript) into trusted websites. The main aim is to steal sensitive information (like session cookies or personal data) from users who visit the compromised website.	is a security vulnerability that occurs when an application directly exposes internal object references (like file paths, database records, or keys) to users without validating proper access rights. An attacker can manipulate these references to access unauthorized resources.			
PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo		Data protection				Input validation
	Descripción de la buena práctica	• Input validation: Validate all user inputs, especially those that will be rendered as HTML. Validate data length, type, format, and range to minimize	• Data Protection API (DPAPI): A built-in feature for encrypting and decrypting sensitive data, available in ASP.NET Core. You can use the IDataProtectionProvide	• Obtain a valid SSL/TLS certificate: Acquire an SSL/TLS certificate from a trusted certificate authority (CA). The certificate should be domain-validated (DV) or organization-	• Use indirect object references: Instead of using direct object references, create indirect references like indexes or mappings that don't reveal actual	Security Headers are HTTP response headers that provide an additional layer of security for your ASP.NET application by controlling specific browser behaviors and reducing the risk of various client-side attacks. X-Content-Type-Options: This	secure coding practices in ASP.NET applications to prevent injection attacks, such as SQL Injection, Cross-Site Scripting (XSS), and XML Injection. Input Validation: Validate user input fields by using whitelist-based validation, ensuring

		the risk of accepting malicious input.
		• Output encoding: Encode potentially unsafe characters in user-generated content, such as HTML and JavaScript, before rendering it on the page. Use HTML encoding functions like Html.Encode() or HttpUtility.HtmlEncode().
		• Content Security Policy (CSP): Create a policy that restricts the browser from executing scripts that don't adhere to the policy. With CSP, define the allowed sources of scripts and other resources, making it difficult for attackers to inject malicious content.
		• Request validation: Activate the built-in request validation feature in ASP.NET that blocks any requests containing potentially unsafe content by default.
			r interface to create an instance of IDataProtector for secure data protection.
			• Symmetric or Asymmetric encryption: Use .NET cryptographic libraries like Aes (symmetric) or RSACryptoServiceProvider (asymmetric) to perform encryption and decryption. Symmetric encryption uses the same key for encryption and decryption, while asymmetric encryption uses a public-private key pair.
			• Secure String: The .NET SecureString class can be used to store sensitive information like passwords in memory, encrypted by default to prevent memory dumping attacks.
				validated (OV), and issued by a well-known CA. Avoid self-signed certificates for production environments.
				• Configure your web server: Configure your web server (IIS, Kestrel, or other) to use the acquired SSL/TLS certificate. Ensure that HTTPS is enabled, and define secure cipher suites and TLS versions.
				• Redirect HTTP to HTTPS: Ensure that all HTTP requests are automatically redirected to HTTPS, using a 301 or 307 redirect.
				• Enable Strict Transport Security (HSTS): Add the Strict-Transport-Security header to enforce HTTPS connections for all future requests. This prevents man-in-the-middle (MITM) attacks that try to downgrade the connection security.
				• Use secure cookies: Configure your application to use secure cookies (with the Secure and HttpOnly flags) to prevent unauthorized
					system-level identifiers. Ensure the mapping is unique per user session and not predictable.
					• Validate access permissions: Always verify that the user has the necessary access rights to perform actions on the specified resources. Check user roles, claims, or access control lists (ACL) to ensure proper authorization.
					• Perform input validation: Validate the user input to prevent injection of unauthorized object references. Reject any unexpected or malformed inputs.
						header prevents MIME-type sniffing by the browser, reducing the risk of downloading and executing malicious content. Set the value to nosniff in your ASP.NET application:
						app.Use(async (context, next) =>
						{
						
						context.Response.Headers.Add("X-Content-Type-Options", "nosniff");
						await next();
						});
						X-Frame-Options: This header controls how your site can be embedded within an iframe, mitigating the risk of clickjacking attacks. Set the value to DENY, SAMEORIGIN, or ALLOW-FROM, depending on your requirements:
						app.Use(async (context, next) =>
						{
						
						context.Response.Headers.Add("X-Frame-Options", "SAMEORIGIN");
						await next();
						});
						X-XSS-Protection: This header helps protect against Cross-Site Scripting (XSS) attacks by providing control over the built-in XSS filter in some browsers. Set the value to 1; mode=block to enable the XSS filter and block suspicious content:
						app.Use(async (context, next) =>
						{
							that only expected and valid input is accepted.
							Use Regular Expressions, data annotations, or custom validation logic to validate input data formats, lengths, and constraints.
							Reject any unexpected or malformed inputs.
							Set up client-side and server-side validation for a more comprehensive defense.
							Output Encoding:
							Encode any user input displayed on your web pages, using encoding functions provided by the .NET framework or libraries like System.Text.Encodings.Web (ASP.NET Core) or Microsoft.Security.Application.Encoder (ASP.NET MVC).
							Utilize Razor syntax, which automatically encodes output by default (@Model.Content), reducing the risk of XSS attacks.
							Ensure proper encoding is applied for different contexts, such as HTML, JavaScript, CSS, or URLs.

				access through cross-site scripting (XSS) or man-in-the-middle (MITM) attacks.		context.Response.Headers.Add("X-XSS-Protection", "1; mode=block"); await next(); });	
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo	No disponible					
	Descripción de la mala práctica						
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Descripción	No disponible					

FS.24	
Datos generales	
Título	Building Secure .NET Applications
Autores	Ioana Baciu
Fecha	26/08/2024
Fuente	www.Infobest.ro

Tipo de publicación (libro, revista, tesis, etc.)	Blog					
Referencia o detalles de la publicación	https://www.infobest.ro/building-secure-net-applications/					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	No disponible	No disponible	No disponible	X	No disponible	No disponible

Datos para la síntesis								
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	No disponible						
	Descripción mecanismo	No disponible						
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	No disponible						
	Descripción	No disponible						
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	No disponible						
	Descripción amenaza	No disponible						
PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo	Role-Based Security	Secure Communication	Data Protection APIS	Authentication and Authorization	Input Validation	Anti-Forgery Tokens	Encrypt Sensitive Data
	Descripción de la buena práctica	This feature enables developers to control access to parts of an application based on the user's role, ensuring that only authorized users can access sensitive	.NET supports secure communication protocols like HTTPS and SSL/TLS, essential for protecting data in transit between clients and servers.	These APIs provide mechanisms for encrypting and decrypting sensitive data, helping protect information stored in	.NET frameworks, such as ASP.NET Identity and .NET Core Identity, offer robust mechanisms for managing user authentication and authorization, including support for multi-factor authentication (MFA)… Multi-Factor Authentication (MFA): Implement MFA to add an extra	Always validate and sanitize user inputs to prevent attacks such as SQL injection and cross-site scripting (XSS). Use parameterized queries and stored procedures to defend against SQL injection.	Use anti-forgery tokens in web forms to prevent cross-site request forgery	Data Protection API (DPAPI): Use DPAPI to encrypt sensitive data stored in configuration files or databases.

		information or perform critical operations.		databases or transmitted over networks.	layer of security, making it harder for attackers to gain unauthorized access. OAuth and OpenID Connect: These protocols provide secure, token-based authentication for web and mobile applications. Role-Based Access Control (RBAC): Design your application with RBAC to ensure users have only the permissions necessary to perform their tasks, minimizing the risk of accidental or malicious actions.		(CSRF) attacks.	
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo	No disponible						
	Descripción de la mala práctica							
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Descripción	No disponible						

Cadena 5

FS.25	
Datos generales	
Título	Enforce HTTPS in ASP.NET Core
Autores	David Galvan and Rick Anderson
Fecha	24/octubre/2024
Fuente	Microsoft.com

Tipo de publicación (libro, revista, tesis, etc.)	Documentación official					
Referencia o detalles de la publicación	https://learn.microsoft.com/en-us/aspnet/core/security/enforcing-ssl?view=aspnetcore-9.0&tabs=visual-studio%2Clinux-sles					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	No disponible	X	No disponible	X	X	No disponible

Datos para la síntesis

PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	No disponible
	Descripción mecanismo	No disponible
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	HTTPS
	Descripción	In some backend service scenarios where connection security is handled at the public-facing edge of the network, configuring connection security at each node isn't required. Web apps that are generated from the templates in Visual Studio or from the dotnet new command enable HTTPS redirection and HSTS. For deployments that don't require these scenarios, you can opt-out of HTTPS/HSTS when the app is created from the template.
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	No disponible
	Descripción amenaza	No disponible
PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo	
	Descripción de la buena práctica	We recommend that production ASP.NET Core web apps use: HTTPS Redirection Middleware (UseHttpsRedirection) to redirect HTTP requests to HTTPS. HSTS Middleware (UseHsts) to send HTTP Strict Transport Security Protocol (HSTS) headers to clients. We recommend using temporary redirects rather than permanent redirects…We recommend using HSTS to signal to clients that only secure resource requests should be sent to the app (only in production). When redirecting to HTTPS without the requirement for additional redirect rules, we recommend using HTTPS Redirection Middleware (UseHttpsRedirection)

		HTTP Strict Transport Security (HSTS) is an opt-in security enhancement that's specified by a web app through the use of a response header
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo	
	Descripción de la mala práctica	Hsts isn't recommended in development because the HSTS settings are highly cacheable by browsers. Do not create a development certificate in an environment that will be redistributed, such as a container image or virtual machine. Doing so can lead to spoofing and elevation of privilege.
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Descripción	No disponible

FS.26						
Datos generales						
Título	6 security best practices for ASP.NET Core					
Autores	Joydip Kanjilal					
Fecha	07/06/2024					
Fuente	www.infoworld.com					
Tipo de publicación (libro, revista, tesis, etc.)	Blog					
Referencia o detalles de la publicación	https://www.infoworld.com/article/2337469/6-security-best-practices-for-aspnet-core.html					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6

	X	No disponible	X	X	No disponible	No disponible

Datos para la síntesis				
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	Enforce HTTPS		
	Descripción mecanismo	SSL, or Secure Sockets Layer, is a protocol that facilitates safe and secure communication between clients and servers over a network by enabling the communication to be encrypted.		
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	No disponible		
	Descripción	No disponible		
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	Cross-site request forgery attacks (CSRF)	Cross-site scripting (XSS)	SQL injection
	Descripción amenaza	trick a user into performing malicious activities while the user is logged into an application. These attacks are most commonly performed by tricking users with phishing emails to lure them to malicious websites, where they use an authenticated user's privileges to steal funds from a victim's bank account	Cross-site scripting (XSS) refers to the act of injecting a malicious script using input or form fields of a web page in your application, with the intent of stealing sensitive data such as login credentials or cookies. When an attacker wants to launch an XSS attack, they often send a malicious link to a user and then attempt to entice the person to click on the link.	SQL injection occurs when an attacker inserts malicious SQL commands within your dynamically created SQL queries. Such attacks are enabled by security vulnerabilities in database queries, leading to exposure of sensitive information.
PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo	HSTS		
	Descripción de la buena práctica	HTTP Strict Transport Security, or HSTS, prevents downgrade protocol attacks and cookie hijacking by ensuring that the web server communicates using an HTTPS connection and by blocking all insecure HTTP connections.	You can protect users of your ASP.NET Core application from CSRF attacks by using anti-forgery tokens. When you include anti-forgery tokens in your application, two different values are sent to the server with each POST. One of the values is sent as a browser cookie, and one is submitted as form data. Unless the server receives both values, it will refuse to allow the request to proceed.	

PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo	No disponible
	Descripción de la mala práctica	
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Descripción	No disponible

FS.27						
Datos generales						
Título	10 Best Practices to Secure ASP.NET Core MVC Web Applications					
Autores	Karthik E					
Fecha	20/11/2024					
Fuente	www.syncfusion.com					
Tipo de publicación (libro, revista, tesis, etc.)	Blog					
Referencia o detalles de la publicación	https://www.syncfusion.com/blogs/post/10-practices-secure-asp-net-core-mvc-app					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	X	No disponible	X	X	X	No disponible
Datos para la síntesis						
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo					
	Descripción mecanismo	SSL stands for Secure Sockets Layer, and it establishes a secure or encrypted connection between client and server. With SSL, the requests passed between the client browser and the server, and the responses from the server to the client browser will be				

		encrypted to maintain the integrity of the data…We can use HTTPS (HyperText Transfer Protocol Secure) to secure your ASP.NET Core application.			
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	No disponible			
	Descripción	No disponible			
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	Cross-Site Scripting (XSS)	SQL Injection	Cross-Site Request Forgery (CSRF)	XXE (XML External Entity) Attack
	Descripción amenaza	Injecting a malicious script through the input/form field of a webpage with the intension to steal confidential information such as login credentials or other authentication information, cookies, and session values is called a cross-site scripting (XSS) attack.	attack wherein unauthorized users inject malicious SQL code that then runs into your database, allowing the attackers to access confidential information stored in it.	An attacker acts as a trusted source and sends some forged data to a site. The site processes the forged data because it believes it is coming from a trusted source.	In this kind of attack, a weakly configured XML parser processes an XML input that contains malicious XML code or a reference to an external entity. This kind of attack can cause a denial-of-service attack by injecting entities within entities, which makes your server utilization too high, resulting in a server shutdown.
PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo				
	Descripción de la buena práctica	Regular Expression Attribute: You can use regular expressions to validate the user's form inputs. So, you can deny malicious characters, and symbols, or allow only acceptable required characters in the input field before allowing the user to proceed further. Regular Expression Object Model: using the regular expression object model, you can validate user inputs by calling static methods of the Regex class. HTML Encoding: The MVC Razor engine automatically encodes all inputs so that the script part provided in any field will never be executed. URL Encoding: we should encode the query parameter input in the URL. Validate Inputs: Validate the user inputs on both the client side and server side. Use Stored Procedures: Using stored procedures will prevent SQL injection, but we should still validate the input parameters passed to the stored procedures.			

		Use Parameterized Queries: you must use parameterized queries to prevent SQL injection. Use Entity Framework or any other ORM: ORM stands for object-relational mapper, which maps SQL objects to your application class object…If you are using Entity Framework properly, you are not prone to SQL injection attacks because Entity Framework internally uses parameterized queries. Store Encrypted Data: We should not store confidential information like email addresses and passwords as plain text in a database. It should be stored in an encrypted format. Prevent Cross-Site Request Forgery: We can prevent this attack by using AntiForgeryToken…We can use the HTML tag helper asp-antiforgery in an HTML attribute and set its value as true. By default, this value will be false. If we set this value as true, it will generate an anti-forgery token. Then, we need to add the [ValidateAntiForgeryToken] attribute to the form post-action method to check whether a valid token is generated. HSTS :HSTS is a web security policy that protects your web application from downgrade protocol attacks and cookie hijacking. It forces the web server to communicate over an HTTPS connection. It always rejects insecure HTTP connections. Prevent XML Attack: If we use XmlTextReader to parse XML files, we must set the DtdProcessing property to Prohibit or Ignore. Authentication Audit: It is a best practice to keep monitoring your production web application's activity logs at regular intervals…Based on the logs, we can gather insights on any errors or performance issues in the production application, and also, if anyone tries to attack the application, we can identify their attempt		
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo			
	Descripción de la mala práctica	It is not recommended for use in the development environment as the browser caches the HSTS header.	We could make mistakes like not removing the authentication cookies after a successful logout.	Do not store sensitive data that includes database or application code anywhere.
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Descripción	No disponible		

FS.28						
Datos generales						
Título	ASP.NET security best practices					
Autores	Tristan Kalos					
Fecha	19/12/2023					
Fuente	www.escape.tech					
Tipo de publicación (libro, revista, tesis, etc.)	Blog					
Referencia o detalles de la publicación	https://escape.tech/blog/asp-dot-net-security/					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	X	No disponible	X	X	No disponible	No disponible

Datos para la síntesis			
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	Enforce HTTPS	Anti-Forgery Tokens
	Descripción mecanismo	Using HTTPS ensures that the data between your server and the client is encrypted, making it more difficult for attackers to intercept or tamper with data.	Anti-forgery tokens prevent Cross-Site Request Forgery (CSRF) attacks by ensuring that the requests sent to your server are from legitimate users.
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	No disponible	
	Descripción	No disponible	
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	CSRF	
	Descripción amenaza	CSRF attacks exploit the trust that a web application has in an authenticated user's browser, allowing attackers to perform unauthorized actions on behalf of the user. For instance, if a user is authenticated in one tab of their browser and visits a malicious site in another, the malicious site can send requests to the authenticated site without the user's knowledge. These requests can change user settings, post content, or even initiate transactions.	

PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo	
	Descripción de la buena práctica	Configuration files often contain sensitive data, such as database connection strings and API keys, which are essential for the application's operations…Encrypting these files adds a robust layer of security, making it much harder for unauthorized individuals to decipher the contents even if they gain access to the files. This practice is a straightforward yet effective way to safeguard important credentials and configuration details, ensuring they remain confidential and secure.
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo	No disponible
	Descripción de la mala práctica	
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Descripción	No disponible

Cadena 6.

FS. 29	
Datos generales	
Título	Storing secrets in web applications using vaults
Autores	Aleksander Młodak
Fecha	19/04/2023
Fuente	Security.pl

Tipo de publicación (libro, revista, tesis, etc.)	Blog					
Referencia o detalles de la publicación	https://www.securing.pl/en/storing-secrets-in-web-applications-using-vaults/					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	No disponible	No disponible	No disponible	X	No disponible	No disponible

Datos para la síntesis		
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	Secrets Management
	Descripción mecanismo	
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	No disponible
	Descripción	No disponible
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	No disponible
	Descripción amenaza	No disponible

PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Descripción de la buena práctica	Take the entire secret lifecycle into account Secrets are created, then rotated (either as a planned action or in response to an incident), and finally revoked or expired. The security of the system can be weakened by neglecting any of these steps Keep an inventory of all secrets It is crucial to know how many secrets are in use, where they live, and who has access to them. Without this basic knowledge, you are about to become a victim of secrets sprawl. Allow access only to authorized clients Enforce access control so that only clients that need a specific secret to do their job are granted access. Log critical operations Because of the sensitive nature of secrets, operations such as accessing or rotating secrets should be logged. Encrypt your secrets Secrets should remain encrypted at rest. When accessed over the network, TLS protection is a must. Ideally, the amount of time a secret is unencrypted should be minimized. Ensure availability There is a lot to cover to have the secrets ready when needed. At the network level, avoid single points of failure and guarantee access to all clients that need it. Make sure you have regular backups that are recoverable.
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Descripción de la mala práctica	Poorly generated secrets will not provide an adequate level of protection.
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Reto	No disponible
	Descripción	No disponible

Cadena 7.

FS. 30	
Datos generales	
Título	10 Best Practices to Secure ASP.NET Core MVC Web Applications

Autores	Karthik E					
Fecha	20/11/2024					
Fuente	syncfusion.com					
Tipo de publicación (libro, revista, tesis, etc.)	Blog					
Referencia o detalles de la publicación	https://www.syncfusion.com/blogs/post/10-practices-secure-asp-net-core-mvc-app#cross-site-request-forgery					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	X	No disponible	X	X	X	No disponible

Datos para la síntesis			
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	AntiForgeryToken	HTTPS Enforcement
	Descripción mecanismo	tag helper asp-antiforgery in an HTML attribute and set its value as true. By default, this value will be false. If we set this value as true, it will generate an anti-forgery token. Then, we need to add the [ValidateAntiForgeryToken] attribute to the form post-action method to check whether a valid token is generated.	HSTS is a web security policy that protects your web application from downgrade protocol attacks and cookie hijacking. It forces the web server to communicate over an HTTPS connection. It always rejects insecure HTTP connections
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	No disponible	No disponible
	Descripción	No disponible	No disponible
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	Cross-Site Request Forgery (CSRF)	
	Descripción amenaza	An attacker acts as a trusted source and sends some forged data to a site. The site processes the forged data because it believes it is coming from a trusted source.	
PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo		
	Descripción de la buena práctica		MaxAge: Timespan that defines the max-age of the Strict-Transport-Security The default value is 30 days.

			IncludeSubDomains: If this value is set to true, the Strict-Transport-Security header will be available for subdomains too. Preload: Adds preload support to the Strict-Transport-Security ExcludedHosts: A list of host names that will not add the HSTS header.
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo		
	Descripción de la mala práctica		The ASP.NET Core template, by default, adds HSTS middleware. It is not recommended for use in the development environment as the browser caches the HSTS header.
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Reto	No disponible	No disponible
	Descripción	No disponible	No disponible

Cadena 8.

FS. 31	
Datos generales	
Título	Enable Cross-Origin Requests (CORS) in ASP.NET Core
Autores	Microsoft, Rick Anderson y Kirk Larkin
Fecha	09/21/2024

Fuente	Microsoft.com					
Tipo de publicación (libro, revista, tesis, etc.)	Documentación official					
Referencia o detalles de la publicación	https://learn.microsoft.com/en-us/aspnet/core/security/cors?view=aspnetcore-9.0					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	X	No disponible	No disponible	X	X	No disponible

Datos para la síntesis		
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	CORS
	Descripción mecanismo	Is a W3C standard that allows a server to relax the same-origin policy. Allows a server to explicitly allow some cross-origin requests while rejecting others.
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	No disponible
	Descripción	No disponible
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	No disponible
	Descripción amenaza	No disponible
PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo	
	Descripción de la buena práctica	UseCors must be called in the correct order. For example, UseCors must be called before UseResponseCaching when using UseResponseCaching. Enabling CORS with the [EnableCors] attribute and applying a named policy to only those endpoints that require CORS provides the finest control. For the finest control of limiting CORS requests: Use [EnableCors("MyPolicy")] with a named policy. Don't define a default policy. Don't use endpoint routing.
PI-5	Mala práctica asociada al mecanismo	

Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Descripción de la mala práctica	Use the [EnableCors] attribute or middleware, not both in the same app. Specifying AllowAnyOrigin and AllowCredentials is an insecure configuration and can result in cross-site request forgery. The CORS service returns an invalid CORS response when an app is configured with both methods. Allowing cross-origin credentials is a security risk. A website at another domain can send a signed-in user's credentials to the app on the user's behalf without the user's knowledge.
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Reto	No disponible
	Descripción	No disponible

FS. 32						
Datos generales						
Título	Implementation and Challenges of CORS in Web Applications Developed with Csharp: A Technical and Practical Analysis					
Autores	Nagib Sabbag Filho					
Fecha	02/09/2024					
Fuente	Leaders.tec.br.com					
Tipo de publicación (libro, revista, tesis, etc.)	Blog					
Referencia o detalles de la publicación	https://leaders.tec.br/article/26e6c8					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	X	No disponible	X	No disponible	No disponible	X

Datos para la síntesis		
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	CORS
	Descripción mecanismo	is a security mechanism that allows restricted resources on a web page to be requested from a different domain than the one that served the page. In other words, CORS defines how a server should allow or restrict access to a resource for a web client from a different origin.

PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	No disponible
	Descripción	No disponible
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	Cross-Site Request Forgery (CSRF)
	Descripción amenaza	
PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo	
	Descripción de la buena práctica	It is important to position the call to UseCors correctly in the pipeline, before middleware that handles requests, such as authentication or endpoints, to ensure that the CORS policy is applied correctly.6
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo	No disponible
	Descripción de la mala práctica	No disponible
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Reto	
	Descripción	common challenge is debugging difficulty, as CORS-related errors may not be clearly reported by browsers, making it hard to identify the source of the problem. Often, when a CORS request fails, the browser simply blocks the request without providing detailed information about the reason for the block . Implementing CORS can also be complicated in scenarios where the application needs to support multiple origins with different permission levels. In such cases, it is important to define specific CORS policies for each origin or group of origins, ensuring that each has the appropriate permissions to access the necessary resources main issues is overly permissive configuration, which can expose the application to security risks

FS. 33	
Datos generales	
Título	What is Cross-Origin Resource Sharing (CORS)?

Autores	Ramotion					
Fecha	03/10/2023					
Fuente	Ramotion.com					
Tipo de publicación (libro, revista, tesis, etc.)	Blog					
Referencia o detalles de la publicación	https://www.ramotion.com/blog/what-is-cors-in-web-development/#section-best-practices-and-potential-vulnerabilities					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	X	No disponible	X	X	X	No disponible

Datos para la síntesis		
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	CORS
	Descripción mecanismo	is a mechanism that allows restricted resources (e.g., fonts, JavaScript, and CSS files) on a web page to be requested from another domain outside the domain from which the resource originated. In other words, it's a communication between domains in different URLs.
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	No disponible
	Descripción	No disponible
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	cross-site scripting (XSS)
	Descripción amenaza	
PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo	
	Descripción de la buena práctica	Set headers appropriately. Set headers such as "Access-Control-Allow-Origin" and "Access-Control-Allow-Headers" appropriately so that you don't accidentally expose sensitive information about your application or users through these requests Use preflight requests. Preflight requests allow your server to determine if it can respond appropriately before actually performing the request. If a preflight request fails, the browser will not send the actual request. This prevents unauthorized cross-domain requests from being made by malicious users or bots that might attempt to abuse your website

PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo	
	Descripción de la mala práctica	Don't use wildcard origin requests. Wildcard origin requests allow you to specify that any origin can make a request, which raises Content Security Policy (CSP) violations by malicious websites or bots attempting to perform cross-domain requests.
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Reto	No disponible
	Descripción	No disponible

FS. 34

Datos generales

Título	Handling CORS (Cross-Origin Resource Sharing) in ASP.NET Core Web API					
Autores	Sardar Mudassar Ali Khan					
Fecha	15/01/2024					
Fuente	C-sharpcorner.com					
Tipo de publicación (libro, revista, tesis, etc.)	Blog					
Referencia o detalles de la publicación	https://www.c-sharpcorner.com/article/handling-cors-cross-origin-resource-sharing-in-asp-net-core-web-api/					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	X	No disponible	X	X	No disponible	No disponible

Datos para la síntesis		
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	CORS
	Descripción mecanismo	is a security feature implemented by web browsers to prevent web pages from making requests to a different domain than the one that served the web page.
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	No disponible
	Descripción	No disponible
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	unauthorized Access
	Descripción amenaza	
PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo	
	Descripción de la buena práctica	Wildcard Origins: Instead of specifying a single origin, you can use Builder.AllowAnyOrigin() to allow requests from any origin. Be cautious with this approach, as it may pose security risks. Credentials: If your API and frontend are on different domains and you need to send credentials (e.g., cookies), consider adding.AllowCredentials() to your CORS policy. Fine-grained Control: You can customize the CORS policy further to allow specific methods, headers, and more.
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo	No disponible
	Descripción de la mala práctica	No disponible
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Reto	No disponible
	Descripción	No disponible

Cadena 9.

FS. 35
Datos generales

Título	Prevent Cross-Site Scripting (XSS) in ASP.NET Core					
Autores	Microsoft, Rick Anderson					
Fecha	09/27/2024					
Fuente	Microsft.com					
Tipo de publicación (libro, revista, tesis, etc.)	Documentación official					
Referencia o detalles de la publicación	https://learn.microsoft.com/en-us/aspnet/core/security/cross-site-scripting?view=aspnetcore-9.0					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	X	No disponible	X	X	X	No disponible

Datos para la síntesis		
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	HTML Encoder
	Descripción mecanismo	Output encoding ensures that any data returned by the API is properly sanitized so that it can't be executed as code by the user's browser.
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	No disponible
	Descripción	No disponible
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	Cross-Site Scripting (XSS)
	Descripción amenaza	is a security vulnerability that enables a cyberattacker to place client side scripts (usually JavaScript) into web pages. When other users load affected pages, the cyberattacker's scripts run, enabling the cyberattacker to steal cookies and session tokens, change the contents of the web page through DOM manipulation, or redirect the browser to another page.
PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo	
	Descripción de la buena práctica	Before putting untrusted data inside an HTML element, ensure it's HTML encoded. HTML encoding takes characters such as < and changes them into a safe form like &lt; Use one of the following approaches to prevent code from being exposed to DOM-based XSS: createElement() and assign property values with appropriate methods or properties such as node.textContent= or node.InnerText=.

		document.CreateTextNode() and append it in the appropriate DOM location. element.SetAttribute() element[attribute]= Validation can be a useful tool in limiting XSS attacks.
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo	
	Descripción de la mala práctica	Never put untrusted data into your HTML input, unless you follow the rest of the steps below. ASP.NET Core MVC provides an HtmlString class which isn't automatically encoded upon output. This should never be used in combination with untrusted input as this will expose an XSS vulnerability Do NOT concatenate untrusted input in JavaScript to create DOM elements or use document.write() on dynamically generated content. Don't use untrusted input as part of a URL path. Always pass untrusted input as a query string value. Never rely on validation alone. Always encode untrusted input before output, no matter what validation or sanitization has been performed.
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Reto	No disponible
	Descripción	No disponible

FS. 36						
Datos generales						
Título	Avoiding Cross-Site Scripting (XSS) attacks in C# and .NET Core					
Autores	Ajay Kumar					
Fecha	04/20/2024					
Fuente	c-sharpcorner.com					
Tipo de publicación (libro, revista, tesis, etc.)	Blog					
Referencia o detalles de la publicación	https://www.c-sharpcorner.com/article/avoiding-cross-site-scripting-xss-attacks-in-c-sharp-and-net-core/					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6

	X	X	X	X	No disponible	No disponible

Datos para la síntesis		
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	HTML Encoder
	Descripción mecanismo	Encode user-generated content before rendering it in HTML to prevent XSS attacks.
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	
	Descripción	Consider a simple web application—a comment section where users can post messages. Without proper validation and sanitization, this application is susceptible to XSS attacks.
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	Cross-Site Scripting (XSS)
	Descripción amenaza	occurs when attackers inject malicious scripts into web pages viewed by other users. These scripts exploit vulnerabilities in the application's handling of user inputs, leading to unauthorized access, data theft, or manipulation.
PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo	
	Descripción de la buena práctica	Input Validation and Sanitization: In C# and .NET Core, validate and sanitize user inputs rigorously to eliminate XSS vulnerabilities. Implementing Content Security Policy (CSP): Configure CSP headers to restrict the execution of scripts from unauthorized sources. Utilizing Anti-Forgery Tokens: Protect against CSRF attacks by using anti-forgery tokens.
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo	No disponible
	Descripción de la mala práctica	No disponible
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Reto	No disponible
	Descripción	No disponible

FS. 37

Datos generales

Título	The Complete Guide to Content Security Policy (CSP) in ASP.NET					
Autores	Atharva					
Fecha	17/09/2024					
Fuente	Atharvasystem.com					
Tipo de publicación (libro, revista, tesis, etc.)	Blog					
Referencia o detalles de la publicación	https://www.atharvasystem.com/the-complete-guide-to-content-security-policy-csp-in-asp-net/					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	X	No disponible	X	X	No disponible	X
Datos para la síntesis						

PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	HTML Encoder
	Descripción mecanismo	
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	No disponible
	Descripción	No disponible
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	Cross-Site Scripting (XSS)
	Descripción amenaza	An attacker injects malicious scripts into a trusted website, leading to data theft and session hijacking. CSP disables the execution of inline scripts and allows only those scripts from trusted sources. Data Injection Attacks – Attackers inject untrusted content like malicious JAvaScript, styles, or images. CSP restricts the load such resources and reduces the risks. Clickjacking—Any malicious site embeds an ASP.NET application within an invisible iframe. It tricks users into acting on the content, making them click a button. CSP will executive directive frame-ancestors and mitigate the clickjacking attack. Mixed Content Vulnerabilities—An HTTPS page will start loading resources over HTTP, exposing the application to a potential attack. CSP will block mixed content and provide stronger security and encryption. Man-in-the-Middle Attack – Attacker will intercept communication between the user and the browser by injecting malicious code. CSP will enforce HTTPS-only connections and will limit the resources from which content gets loaded.

PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo	CSP
	Descripción de la buena práctica	set up a robust CSP policy. With a CSP in place, you can prevent XSS attacks, data injection and clickjacking. you must ensure to adopt is monitoring CSP violations and improving policy enforcement with report-uri directives. This directive will specify the place where the browser must send reports at the time of the breach. test and redefine your CSP policy from time to time.
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo	No disponible
	Descripción de la mala práctica	No disponible
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Reto	
	Descripción	CSP can block legitimate resources such as scripts, fonts, styles, or images if they are loaded from third-party services. To prevent this issue, you can whitelist trusted external sources in the CSP directives. CSP may also block inline JavaScript and CSS by default. This issue may occur with a common attack vector for XSS. The solution is to not allow 'unsafe-inline' and use hashes to get that trusted inline code. The process of understanding and debugging CSP is a time-consuming activity. In this case to cut down on unnecessary time, you can start with the report-only mode to collect violations. You can refine the policy iteratively. Sometimes, CSP inadvertently blocks dynamic content loaded via JavaScript, which may break the functionality. To address these challenges, you can adjust the CSP policy by explicitly allowing trusted sources for data connections. All browsers do not support CSP enforcement. Hence, the solution is to develop according to the modern browsers and their compatibility. Maintaining a rigorous CSP policy becomes overwhelming as the application grows and starts integrating third-party libraries and services. To solve this problem, you can develop automated testing that identifies CSP violations early.

FS. 38	
Datos generales	
Título	ASP.NET Website Hosting: Best Practices For Secure Your Web Applications

Autores	Ajay Kumar					
Fecha	10/10/2024					
Fuente	accuwebhosting.com					
Tipo de publicación (libro, revista, tesis, etc.)	Blog					
Referencia o detalles de la publicación	https://www.accuwebhosting.com/blog/asp-net-website-hosting-best-practices-for-secure-your-web-applications/					
Pregunta de investigación que responde	PI-1	PI-2	PI-3	PI-4	PI-5	PI-6
	X	No disponible	X	X	No disponible	No disponible

Datos para la síntesis		
PI-1 Mecanismos de seguridad que provee ASP.NET Core	Nombre mecanismo	HTML Encoder
	Descripción mecanismo	Use frameworks like the Razor engine in MVC, which automatically encodes input from variables to prevent script execution
PI-2 Contextos o escenarios donde se utilizan los mecanismos de seguridad en ASP.NET Core	Escenario	No disponible
	Descripción	No disponible
PI-3 Amenazas y/o vulnerabilidad están asociadas a cada uno de los mecanismos de seguridad en ASP.NET Core	Nombre de la amenaza asociada al mecanismo	Cross-Site Scripting (XSS)
	Descripción amenaza	Attackers can insert malicious scripts, frequently written in JavaScript, into online sites by using a security flaw known as cross-site scripting (XSS). When users access these compromised pages, malicious scripts launch immediately, giving hackers the ability to alter the content of the page or take advantage of cookies and session tokens to obtain login credentials.
PI-4 Buenas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Buena práctica asociada al mecanismo	
	Descripción de la buena práctica	Regular Expression Attributes: Validate user inputs using regular expressions to ensure they meet specific criteria. URL Encoding: To minimize XSS risks, avoid using plain text in query strings; instead, use encoded query strings. Regular Expression Object Model: Use static Regex class methods to validate user inputs effectively. Use parameterized queries and stored procedures to stop SQL injection attacks and avoid outdated or risky components and libraries.

		It’s also essential to keep your ASP.NET Core framework and its dependencies updated with the latest patches.
PI-5 Malas prácticas que se asocian a los mecanismos de seguridad en ASP.NET Core	Mala práctica asociada al mecanismo	No disponible
	Descripción de la mala práctica	No disponible
PI-6 Retos que plantea el uso de los mecanismos de seguridad identificados en ASP.NET Core	Reto	No disponible
	Descripción	No disponible