.NET Core Supported Rules

The following rules are supported for .NET Core projects:

- BD.API.EQNULL - Make sure implementation of Object.Equals(Object) properly handles null values
- BD.API.EQREFL - Make sure implementation of Object.Equals(Object) is reflexive
- BD.CO.ITMOD - Do not modify collection while iterating over it
- BD.EXCEPT.AN - Avoid ArgumentNullException
- BD.EXCEPT.NR - Avoid NullReferenceException
- BD.PB.ARRAY - Avoid accessing arrays out of bounds
- BD.PB.CC - Avoid conditions that always evaluate to the same value
- BD.PB.CHECKRET - Consistently check the returned value of non-void methods
- BD.PB.DEREF - Avoid dereferencing before checking for null
- BD.PB.DISP - Do not use disposed resources
- BD.PB.EVIPT - Ensure that invoke parameter type correspond to delegate definition
- BD.PB.FIN - Do not use managed resources in finalizers
- BD.PB.INTOVERF - Avoid integer overflows
- BD.PB.NOTEXPLINIT - Avoid use before explicit initialization
- BD.PB.POVR - Avoid overwriting method parameters before each use
- BD.PB.SBONE - Ensure proper usage of StringBuilder objects
- BD.PB.STRNULL - Do not append null value to strings
- BD.PB.SWITCH - Avoid switch with unreachable branches
- BD.PB.UGHC - Do not base equality on the equality of hash codes
- BD.PB.VOVR - Avoid unused values
- BD.PB.ZERO - Avoid division by zero
- BD.RES.LEAKS - Ensure resources are deallocated
- BD.SECURITY.AUTH - Prevent untrusted inputs that may affect authorization
- BD.SECURITY.CUSTOM - Prevent security vulnerability (custom rule)
- BD.SECURITY.SALT - Ensure that a random salt is used
- BD.SECURITY.SENS - Prevent exposure of sensitive data
- BD.SECURITY.TDCMD - Protect against command injection
- BD.SECURITY.TDFNAMES - Protect against file name injection
- BD.SECURITY.TDINPUT - Exclude unsanitized user input from format strings
- BD.SECURITY.TDLDAP - Protect against LDAP injection
- BD.SECURITY.TDLOG - Protect against log forging
- BD.SECURITY.TDNET - Protect against network resource injection
- BD.SECURITY.TDRESP - Protect against HTTP response splitting
- BD.SECURITY.TDRFL - Protect against Reflection injection
- BD.SECURITY.TDSQL - Protect against SQL query injection
- BD.SECURITY.TDSQLC - Protect against SQL connection injection
- BD.SECURITY.TDXSS - Protect against XSS vulnerabilities
- BD.SECURITY.USXRS - Use object with secure XmlResolver property
- BD.TRS.DIFCS - Variable should be used in context of single critical section
- BD.TRS.INSTLOCK - Do not use an instance lock to protect shared static data
- BD.TRS.MUTEX - Do not abandon unreleased mutexes
- BD.TRS.ORDER - Do not acquire locks in different order
- BRM.CMT.MSC - Members should be commented
- BRM.CMT.TSC - Types should be commented
- BRM.HBCM - Avoid hiding methods from base classes
- BRM.HBCP - Avoid hiding properties from base classes
- BRM.LINUPPERCASE - Use 'L' instead of 'l' to express 'long' integer constants
- BRM.SFH - Always provide appropriate file header (copyright information, etc.)
- BRM.SFT - Use spaces for tabs/indentation
- CDD.DUPC - Avoid code duplication
- CDD.DUPM - Avoid method duplication
- CDD.DUPS - Avoid string literal duplication
- CDD.DUPT - Avoid types duplication
- CS.BRM.AIBA - Avoid 'is' before 'as'
- CS.BRM.BEB - Avoid block statements with empty bodies.
- CS.BRM.CCB - Always enclose if and else bodies with curly braces
- CS.BRM.CPEB - Avoid checked, unchecked, fixed and unsafe statements with empty bodies.
- CS.BRM.ES - Use string.Empty for empty strings
- CS.BRM.ETK - Use keyword 'this' explicitly when accessing capitalized members
- CS.BRM.ICB - Always enclose iteration statements bodies with curly braces.
- CS.BRM.IDOU - Avoid increment/decrement operators inside other expressions.
- CS.BRM.IEB - Avoid initialization statements with empty bodies.
- CS.BRM.KFATNC - Keep file and type names consistent
- CS.BRM.RFINE - Do not use large if-clause with small else-clause that returns.
- CS.BRM.SCHR - Avoid using the Strings.Chr() and Strings.ChrW() methods in C# code.
- CS.BRM.SWDEFLAST - Place 'default' as the last case of the 'switch' statement
- CS.BRM.UCB - Always enclose using statement body with curly braces.
- CS.BRM.UCO - Use null-coalescing operator ('??') instead conditional operator ('?')
- CS.CDD.DUPU - Avoid duplicate using statements
- CS.CMUG.PRU.FSPP - Follow standard pattern for property accessors.
- CS.EU.VZS - Ensure that each enum has member with value 0
- CS.INTER.ITT - String literals should be internationalized
- CS.MLC - Avoid using very large methods
- CS.NG.VAR.PNCFV - Follow proper naming convention for method variables and consts.

- CS.OOM.MI - Keep Maintainability Index above specified value.
- CS.PB.ANIL - Avoid non-iterable loops.
- CS.PB.AWC - Avoid assignment within a condition.
- CS.PB.BITBOOL - Do not use bitwise operators on bool operands.
- CS.PB.CCA - Avoid confusing assignments to constructor arguments.
- CS.PB.CEB - Avoid conditional statements with empty bodies.
- CS.PB.CNFA - Check for 'null' when using 'as' operator
- CS.PB.DEFSWITCH - Provide 'default:' for each 'switch' statement.
- CS.PB.EEB - Avoid try, catch, finally and using statements with empty bodies.
- CS.PB.FPLI - Do not use floating point variables as loop indices.
- CS.PB.IDNE - Avoid increment and decrement statements which have no effect
- CS.PB.IEB - Avoid iteration statements with empty bodies.
- CS.PB.IVFLC - Use initializer variable in a condition of 'for' loop
- CS.PB.IVFLI - Use initializer variable in 'for' loop iterator section.
- CS.PB.MCO - Review '?:' operator for potential misuse.
- CS.PB.NACC - Avoid inaccessible classes and structs.
- CS.PB.NSIVFLB - Do not modify 'for' loop initializer variable in 'for' loop body.
- CS.PB.NSIVFLN - Do not increment or decrement on the same variable over multiple nested 'for' loop statements.
- CS.PB.PUO - Avoid using the unary + operator
- CS.PB.USC.CC - Avoid unreachable code in condition
- CS.PB.USC.UC - Avoid unreachable code
- CS.PB.VTNV - Do not compare value types to null
- CS.PB.WIBS - Avoid wrong indentation of blockless statements
- CS.PE.VFFP - Verify FileDialog filter pattern.
- CS.PFEL - Use foreach loops instead of for loops
- CS.PROTC - Prefer readonly to const
- CS.SC - Cast only simple expressions
- CS.SEC.AUK - Avoid 'unsafe' keyword.
- CS.SERIAL.IIDC - Implement IDeserializationCallback for classes with NonSerialized fields.
- CS.SERIAL.SOIS - Do not store non-serializable objects in Session.
- CS.SERIAL.UIS - Use the standard pattern while implementing ISerializable.
- CS.TRS.LCB - Always enclose lock statement body with curly braces.
- CS.TRS.LEB - Avoid lock statements with empty bodies.
- CS.USO - Put using statements in alphabetical order
- CT.ECLSII - Avoid explicit conversions of integrals to integrals of smaller size if the conversion may cause data truncation
- CT.ECLTS - Avoid explicit conversions between data types if the conversion may cause data loss or unexpected results
- CWE.119.ARRAY - Avoid accessing arrays out of bounds
- CWE.120.AUK - Avoid 'unsafe' keyword.
- CWE.125.ARRAY - Avoid accessing arrays out of bounds
- CWE.129.ARRAY - Avoid accessing arrays out of bounds
- CWE.131.AUK - Avoid 'unsafe' keyword.
- CWE.134.TDINPUT - Exclude unsanitized user input from format strings
- CWE.190.AIWIL - Avoid indexer wraparound in loops.
- CWE.190.INTOVERF - Avoid integer overflows
- CWE.191.AIWIL - Avoid indexer wraparound in loops.
- CWE.191.INTOVERF - Avoid integer overflows
- CWE.197.ECLSII - Avoid explicit conversions of integrals to integrals of smaller size if the conversion may cause data truncation
- CWE.20.ARRAY - Avoid accessing arrays out of bounds
- CWE.20.TDCMD - Protect against command injection
- CWE.20.TDFNAMES - Protect against file name injection
- CWE.20.TDNET - Protect against network resource injection
- CWE.20.TDRESP - Protect against HTTP response splitting
- CWE.20.TDSQL - Protect against SQL query injection
- CWE.20.TDSQLC - Protect against SQL connection injection
- CWE.20.TDXSS - Protect against XSS vulnerabilities
- CWE.200.SENS - Prevent exposure of sensitive data
- CWE.209.SENS - Prevent exposure of sensitive data
- CWE.22.TDFNAMES - Protect against file name injection
- CWE.250.AUEP - Avoid using elevated privileges.
- CWE.252.CHECKRET - Consistently check the returned value of non-void methods
- CWE.285.TDSQL - Protect against SQL query injection
- CWE.287.AAM - Add authorization services to MVC Core
- CWE.287.IIPHEU - Do not rely on reverse DNS resolution for security decisions
- CWE.287.LUAFLA - Lock out the user after failed login attempts
- CWE.287.UAAMC - Ensure that authorization attributes match the controller
- CWE.295.DNICV - Do not disable SSL certificate validation
- CWE.307.LUAFLA - Lock out the user after failed login attempts
- CWE.327.ACCA - Avoid using custom cryptographic algorithms.
- CWE.329.ACCA - Avoid using custom cryptographic algorithms.
- CWE.350.IIPHEU - Do not rely on reverse DNS resolution for security decisions
- CWE.352.TDRESP - Protect against HTTP response splitting
- CWE.352.VAFT - Use anti-forgery attributes on POST methods
- CWE.362.DIFCS - Variable should be used in context of single critical section
- CWE.369.ZERO - Avoid division by zero
- CWE.391.LGE - Ensure all exceptions are either logged with a standard logger or rethrown.
- CWE.396.NCSAE - Avoid the use of "catch" on 'Exception', 'SystemException' or 'ApplicationException'
- CWE.397.NTSAE - Avoid throwing 'Exception', 'SystemException' or 'ApplicationException'
- CWE.400.LEAKS - Ensure resources are deallocated
- CWE.401.DDFODB - In 'Dispose(bool)' use input parameter to check if it is actually disposing.

- CWE.416.DISP - Do not use disposed resources
- CWE.416.FIN - Do not use managed resources in finalizers
- CWE.434.TDFNAMES - Protect against file name injection
- CWE.456.NOTEXPLINIT - Avoid use before explicit initialization
- CWE.457.NOTEXPLINIT - Avoid use before explicit initialization
- CWE.470.TDRFL - Protect against Reflection injection
- CWE.476.CNFA - Check for 'null' when using 'as' operator
- CWE.476.DEREF - Avoid dereferencing before checking for null
- CWE.476.NR - Avoid NullReferenceException
- CWE.480.PUO - Avoid using the unary + operator
- CWE.481.AWC - Avoid assignment within a condition.
- CWE.494.IREC - Do not execute external code without integrity check.
- CWE.502.IIDC - Implement IDeserializationCallback for classes with NonSerialized fields.
- CWE.502.UIS - Use the standard pattern while implementing ISerializable.
- CWE.546.TODO - Ensure that comments do not contain task tags
- CWE.561.UC - Avoid unreachable code
- CWE.563.POVR - Avoid overwriting method parameters before each use
- CWE.563.VOVR - Avoid unused values
- CWE.570.CC - Avoid conditions that always evaluate to the same value
- CWE.571.CC - Avoid conditions that always evaluate to the same value
- CWE.595.REVT - Do not use ReferenceEquals() on value types.
- CWE.601.TDNET - Protect against network resource injection
- CWE.611.PDTDP - Prevent DTD processing
- CWE.611.USXRS - Use object with secure XmlResolver property
- CWE.613.ISE - Ensure sufficient session expiration
- CWE.662.DIFCS - Variable should be used in context of single critical section
- CWE.676.APDM - Avoid using potentially dangerous methods.
- CWE.681.ECLTS - Avoid explicit conversions between data types if the conversion may cause data loss or unexpected results
- CWE.704.ECLTS - Avoid explicit conversions between data types if the conversion may cause data loss or unexpected results
- CWE.759.SALT - Ensure that a random salt is used
- CWE.760.SALT - Ensure that a random salt is used
- CWE.77.TDCMD - Protect against command injection
- CWE.770.LEAKS - Ensure resources are deallocated
- CWE.772.LEAKS - Ensure resources are deallocated
- CWE.78.TDCMD - Protect against command injection
- CWE.780.UOWR - Use OAEP with RSA algorithm encryption.
- CWE.787.ARRAY - Avoid accessing arrays out of bounds
- CWE.79.TDRESP - Protect against HTTP response splitting
- CWE.79.TDXSS - Protect against XSS vulnerabilities
- CWE.80.TDRESP - Protect against HTTP response splitting
- CWE.807.AUTH - Prevent untrusted inputs that may affect authorization
- CWE.827.PDTDP - Prevent DTD processing
- CWE.833.ORDER - Do not acquire locks in different order
- CWE.835.IVFLC - Use initializer variable in a condition of 'for' loop
- CWE.835.IVFLI - Use initializer variable in 'for' loop iterator section.
- CWE.835.NSIVFLN - Do not increment or decrement on the same variable over multiple nested 'for' loop statements.
- CWE.838.AIHUE - Avoid using improper HTML or URL encoding in HttpResponse methods
- CWE.862.UAA - Use authorization attributes on pages and controllers
- CWE.863.AAM - Add authorization services to MVC Core
- CWE.863.AUTH - Prevent untrusted inputs that may affect authorization
- CWE.863.UAAMC - Ensure that authorization attributes match the controller
- CWE.88.TDCMD - Protect against command injection
- CWE.89.TDSQL - Protect against SQL query injection
- CWE.89.TDSQLC - Protect against SQL connection injection
- CWE.90.TDLDAP - Protect against LDAP injection
- CWE.99.TDFNAMES - Protect against file name injection
- CWE.99.TDNET - Protect against network resource injection
- EXCEPT.NCSAE - Avoid the use of "catch" on 'Exception', 'SystemException' or 'ApplicationException'
- EXCEPT.NTSAE - Avoid throwing 'Exception', 'SystemException' or 'ApplicationException'
- IFD.DDFODB - In 'Dispose(bool)' use input parameter to check if it is actually disposing.
- IFD.IDDR - Implement IDisposable in types which are using disposable resources
- INTER.RI - Make sure that all string's from *.resx files are internationalized
- OPU.CPTEQ - Compare parameter type of Equals(Object) method
- OPU.REVT - Do not use ReferenceEquals() on value types.
- OWASP2017.A1.TDCMD - Protect against command injection
- OWASP2017.A1.TDFNAMES - Protect against file name injection
- OWASP2017.A1.TDINPUT - Exclude unsanitized user input from format strings
- OWASP2017.A1.TDLDAP - Protect against LDAP injection
- OWASP2017.A1.TDNET - Protect against network resource injection
- OWASP2017.A1.TDRFL - Protect against Reflection injection
- OWASP2017.A1.TDSQL - Protect against SQL query injection
- OWASP2017.A1.TDSQLC - Protect against SQL connection injection
- OWASP2017.A10.LGE - Ensure all exceptions are either logged with a standard logger or rethrown.
- OWASP2017.A2.ISE - Ensure sufficient session expiration
- OWASP2017.A2.LUAFLA - Lock out the user after failed login attempts
- OWASP2017.A3.ACCA - Avoid using custom cryptographic algorithms.
- OWASP2017.A3.SALT - Ensure that a random salt is used
- OWASP2017.A3.UOWR - Use OAEP with RSA algorithm encryption.
- OWASP2017.A4.PDTDP - Prevent DTD processing

- OWASP2017.A4.USXRS - Use object with secure XmlResolver property
- OWASP2017.A5.AAM - Add authorization services to MVC Core
- OWASP2017.A5.AUEP - Avoid using elevated privileges.
- OWASP2017.A5.UAA - Use authorization attributes on pages and controllers
- OWASP2017.A5.UAAMC - Ensure that authorization attributes match the controller
- OWASP2017.A5.VAFT - Use anti-forgery attributes on POST methods
- OWASP2017.A6.NCSAE - Avoid the use of "catch" on 'Exception', 'SystemException' or 'ApplicationException'
- OWASP2017.A6.NTSAE - Avoid throwing 'Exception', 'SystemException' or 'ApplicationException'
- OWASP2017.A6.SENS - Prevent exposure of sensitive data
- OWASP2017.A7.TDRESP - Protect against HTTP response splitting
- OWASP2017.A7.TDXSS - Protect against XSS vulnerabilities
- OWASP2017.A8.IIDC - Implement IDeserializationCallback for classes with NonSerialized fields.
- OWASP2017.A8.UIS - Use the standard pattern while implementing ISerializable.
- PB.ACDE - Avoid calling the Application.DoEvents() method
- PB.AIHUE - Avoid using improper HTML or URL encoding in HttpResponse methods
- PB.CFF - Verify number of arguments in Composite Formatting feature
- PB.DNCF - Do not compare floating-point types for equality.
- PB.EMPTYMETHODS - Avoid empty methods
- PB.II.TODO - Ensure that comments do not contain task tags
- PB.INOE - Use String.IsNullOrEmpty to check if a string is null or empty
- PB.STATICFLD - Do not write to static fields from non-static methods
- PB.THROWFIN - Avoid 'throw' statements in 'finally' blocks.
- PCIDSS32.651.TDCMD - Protect against command injection
- PCIDSS32.651.TDFNAMES - Protect against file name injection
- PCIDSS32.651.TDINPUT - Exclude unsanitized user input from format strings
- PCIDSS32.651.TDLDAP - Protect against LDAP injection
- PCIDSS32.651.TDNET - Protect against network resource injection
- PCIDSS32.651.TDRFL - Protect against Reflection injection
- PCIDSS32.651.TDSQL - Protect against SQL query injection
- PCIDSS32.651.TDSQLC - Protect against SQL connection injection
- PCIDSS32.6510.ISE - Ensure sufficient session expiration
- PCIDSS32.6510.LUAFLA - Lock out the user after failed login attempts
- PCIDSS32.652.AUK - Avoid 'unsafe' keyword.
- PCIDSS32.653.ACCA - Avoid using custom cryptographic algorithms.
- PCIDSS32.653.SALT - Ensure that a random salt is used
- PCIDSS32.653.UOWR - Use OAEP with RSA algorithm encryption.
- PCIDSS32.655.CHECKRET - Consistently check the returned value of non-void methods
- PCIDSS32.655.LGE - Ensure all exceptions are either logged with a standard logger or rethrown.
- PCIDSS32.655.NCSAE - Avoid the use of "catch" on 'Exception', 'SystemException' or 'ApplicationException'
- PCIDSS32.655.NTSAE - Avoid throwing 'Exception', 'SystemException' or 'ApplicationException'
- PCIDSS32.655.SENS - Prevent exposure of sensitive data
- PCIDSS32.657.TDRESP - Protect against HTTP response splitting
- PCIDSS32.657.TDXSS - Protect against XSS vulnerabilities
- PCIDSS32.658.AAM - Add authorization services to MVC Core
- PCIDSS32.658.AUEP - Avoid using elevated privileges.
- PCIDSS32.658.UAA - Use authorization attributes on pages and controllers
- PCIDSS32.658.UAAMC - Ensure that authorization attributes match the controller
- PCIDSS32.659.TDRESP - Protect against HTTP response splitting
- PCIDSS32.659.VAFT - Use anti-forgery attributes on POST methods
- SEC.ACCA - Avoid using custom cryptographic algorithms.
- SEC.ACWNS - There should be no classes without namespace.
- SEC.AIWIL - Avoid indexer wraparound in loops.
- SEC.APDM - Avoid using potentially dangerous methods.
- SEC.AUEP - Avoid using elevated privileges.
- SEC.IREC - Do not execute external code without integrity check.
- SEC.LGE - Ensure all exceptions are either logged with a standard logger or rethrown.
- SEC.UOWR - Use OAEP with RSA algorithm encryption.
- SEC.WEB.AAM - Add authorization services to MVC Core
- SEC.WEB.DNICV - Do not disable SSL certificate validation
- SEC.WEB.IIPHEU - Do not rely on reverse DNS resolution for security decisions
- SEC.WEB.ISE - Ensure sufficient session expiration
- SEC.WEB.LUAFLA - Lock out the user after failed login attempts
- SEC.WEB.UAA - Use authorization attributes on pages and controllers
- SEC.WEB.UAAMC - Ensure that authorization attributes match the controller
- SEC.WEB.VAFT - Use anti-forgery attributes on POST methods
- SEC.XXE.PDTDP - Prevent DTD processing
- SERIAL.XML.SOAFAP - Make sure that type of serialized field/property is compatible with type used in XmlElementAttribute
- VB.BRM.ES - Use string.Empty for empty strings
- VB.BRM.PNPT - Use preferred names for primitive types
- VB.PB.DEFSWITCH - Provide 'Case Else' for each 'Select Case' statement