

ISO/IEC TS 17961 C SECURE CODING RULES
TECHNICAL SPECIFICATION CATEGORIES
MAPPED TO CODESONAR[®] 7.4



INTRODUCTION

ISO/IEC TS 17961 specifies rules for secure coding in the C programming language.

For more information on ISO/IEC TS 17961:

<https://www.iso.org/standard/61134.html>

The remainder of this document comprises two tables:

- A table showing the close mapping between CodeSonar warning classes and the ISO/IEC TS 17961 categories.
- A table showing the broad mapping between CodeSonar warning classes and the ISO/IEC TS 17961 categories. The broad mapping for a CodeSonar warning class includes the close mapping for the class, plus any other checks that are related to the class in a meaningful way, but not eligible for the close mapping.



ISO/IEC TS 17961 CLOSE MAPPING (CODESONAR V7.4)

The following table contains CodeSonar warning classes that are closely mapped to ISO/IEC TS 17961 categories.

Rule	Rule Name	Supported
TS17961:5.1-ptrcomp	5.1. Accessing an object through a pointer to an incompatible type	Yes
TS17961:5.2-accfree	5.2. Accessing freed memory	Yes
TS17961:5.3-accsig	5.3. Accessing shared objects in signal handlers	Yes
TS17961:5.4-boolasgn	5.4. No assignment in conditional expressions	Yes
TS17961:5.5-asyncsig	5.5. Calling functions in the C Standard Library other than abort, _Exit, and signal from within a signal handler	Yes
TS17961:5.6-argcomp	5.6. Calling functions with incorrect arguments	Yes
TS17961:5.7-sigcall	5.7. Calling signal from interruptible signal handlers	Yes
TS17961:5.8-syscall	5.8. Calling system	Yes
TS17961:5.9-padcomp	5.9. Comparison of padding data	No
TS17961:5.10-intptrconv	5.10. Converting a pointer to integer or integer to pointer	Yes
TS17961:5.11-alignconv	5.11. Converting pointer values to more strictly aligned pointer types	No
TS17961:5.12-filecpy	5.12. Copying a FILE object	Yes
TS17961:5.13-funcdecl	5.13. Declaring the same function or object in incompatible ways	Yes
TS17961:5.14-nullref	5.14. Dereferencing an out-of-domain pointer	Yes
TS17961:5.15-addresscape	5.15. Escaping of the address of an automatic object	Yes
TS17961:5.16-signconv	5.16. Conversion of signed characters to wider integer types before a check for EOF	Yes
TS17961:5.17-swtdchflt	5.17. Use of an implied default in a switch statement	Yes
TS17961:5.18-fileclose	5.18. Failing to close files or free dynamic memory when they are no longer needed	Yes
TS17961:5.19-liberr	5.19. Failing to detect and handle standard library errors	Yes
TS17961:5.20-libptr	5.20. Forming invalid pointers by library function	No
TS17961:5.21-invptr	5.21. Forming or using out-of-bounds pointers or array subscripts	Yes
TS17961:5.22-dblfree	5.22. Freeing memory multiple times	Yes
TS17961:5.23-usrfmt	5.23. Including tainted or out-of-domain input in a format string	Yes
TS17961:5.24-inverno	5.24. Incorrectly setting and using errno	Yes
TS17961:5.25-diverr	5.25. Integer division errors	Yes
TS17961:5.26-ioileave	5.26. Interleaving stream inputs and outputs without a flush or positioning call	No
TS17961:5.27-strmod	5.27. Modifying string literals	Yes
TS17961:5.28-libmod	5.28. Modifying the string returned by getenv, localeconv, setlocale, and strerror	Yes
TS17961:5.29-intoflow	5.29. Overflowing signed integers	Yes
TS17961:5.30-nonnullstr	5.30. Passing a non-null-terminated string to a library function	Yes
TS17961:5.31-chrsgnext	5.31. Passing arguments to character-handling functions that are not representable as unsigned char	Yes
TS17961:5.32-restrict	5.32. Passing pointers into the same object as arguments to different restrict-qualified parameters	Yes
TS17961:5.33-xfree	5.33. Reallocating or freeing memory that was not dynamically allocated	Yes
TS17961:5.34-uninitref	5.34. Referencing uninitialized memory	Yes
TS17961:5.35-ptobj	5.35. Subtracting or comparing two pointers that do not refer to the same array	Yes
TS17961:5.36-taintstcpy	5.36. Tainted strings are passed to a string copying function	Yes
TS17961:5.37-sizeofptr	5.37. Taking the size of a pointer to determine the size of the pointed-to type	Yes
TS17961:5.38-taintnoproto	5.38. Using a tainted value as an argument to an unprototyped function pointer	Yes
TS17961:5.39-taintformatio	5.39. Using a tainted value to write to an object using a formatted input or output function	Yes

TS17961:5.40-xfilepos	5.40. Using a value for fsetpos other than a value returned from fgetpos	No
TS17961:5.41-libuse	5.41. Using an object overwritten by getenv, localeconv, setlocale, and strerror	No
TS17961:5.42-chreof	5.42. Using character values that are indistinguishable from EOF	No
TS17961:5.43-resident	5.43. Using identifiers that are reserved for the implementation	No
TS17961:5.44-infmtstr	5.44. Using invalid format strings	Yes
TS17961:5.45-taintsink	5.45. Tainted, potentially mutilated, or out-of-domain integer values are used in a restricted sink	Yes



ISO/IEC TS 17961 BROAD MAPPING (CODESONAR V7.4)

The following table contains CodeSonar warning classes that are broadly mapped to ISO/IEC TS 17961 categories.

Rule	Rule Name	Supported
TS17961:5.1-ptrcomp	5.1. Accessing an object through a pointer to an incompatible type	Yes
TS17961:5.2-accfree	5.2. Accessing freed memory	Yes
TS17961:5.3-accsig	5.3. Accessing shared objects in signal handlers	Yes
TS17961:5.4-boolasgn	5.4. No assignment in conditional expressions	Yes
TS17961:5.5-asynsig	5.5. Calling functions in the C Standard Library other than abort, _Exit, and signal from within a signal handler	Yes
TS17961:5.6-argcomp	5.6. Calling functions with incorrect arguments	Yes
TS17961:5.7-sigcall	5.7. Calling signal from interruptible signal handlers	Yes
TS17961:5.8-syscall	5.8. Calling system	Yes
TS17961:5.9-padcomp	5.9. Comparison of padding data	Yes
TS17961:5.10-intptrconv	5.10. Converting a pointer to integer or integer to pointer	Yes
TS17961:5.11-alignconv	5.11. Converting pointer values to more strictly aligned pointer types	No
TS17961:5.12-filecpy	5.12. Copying a FILE object	Yes
TS17961:5.13-funcdecl	5.13. Declaring the same function or object in incompatible ways	Yes
TS17961:5.14-nullref	5.14. Dereferencing an out-of-domain pointer	Yes
TS17961:5.15-addresscape	5.15. Escaping of the address of an automatic object	Yes
TS17961:5.16-signconv	5.16. Conversion of signed characters to wider integer types before a check for EOF	Yes
TS17961:5.17-swtdhflt	5.17. Use of an implied default in a switch statement	Yes
TS17961:5.18-fileclose	5.18. Failing to close files or free dynamic memory when they are no longer needed	Yes
TS17961:5.19-liberr	5.19. Failing to detect and handle standard library errors	Yes
TS17961:5.20-libptr	5.20. Forming invalid pointers by library function	Yes
TS17961:5.21-invptr	5.21. Forming or using out-of-bounds pointers or array subscripts	Yes
TS17961:5.22-dblfree	5.22. Freeing memory multiple times	Yes
TS17961:5.23-usrfmt	5.23. Including tainted or out-of-domain input in a format string	Yes
TS17961:5.24-inverno	5.24. Incorrectly setting and using errno	Yes
TS17961:5.25-diverr	5.25. Integer division errors	Yes
TS17961:5.26-ioileave	5.26. Interleaving stream inputs and outputs without a flush or positioning call	Yes
TS17961:5.27-strmod	5.27. Modifying string literals	Yes
TS17961:5.28-libmod	5.28. Modifying the string returned by getenv, localeconv, setlocale, and strerror	Yes
TS17961:5.29-intoflow	5.29. Overflowing signed integers	Yes
TS17961:5.30-nonnullstr	5.30. Passing a non-null-terminated string to a library function	Yes
TS17961:5.31-chrsgnext	5.31. Passing arguments to character-handling functions that are not representable as unsigned char	Yes
TS17961:5.32-restrict	5.32. Passing pointers into the same object as arguments to different restrict-qualified parameters	Yes
TS17961:5.33-xfree	5.33. Reallocating or freeing memory that was not dynamically allocated	Yes
TS17961:5.34-uninitref	5.34. Referencing uninitialized memory	Yes
TS17961:5.35-ptobj	5.35. Subtracting or comparing two pointers that do not refer to the same array	Yes
TS17961:5.36-taintstrep	5.36. Tainted strings are passed to a string copying function	Yes
TS17961:5.37-sizeofptr	5.37. Taking the size of a pointer to determine the size of the pointed-to type	Yes
TS17961:5.38-taintnproto	5.38. Using a tainted value as an argument to an unprototyped function pointer	Yes
TS17961:5.39-taintformatio	5.39. Using a tainted value to write to an object using a formatted input or output function	Yes



TS17961:5.40-xfilepos	5.40. Using a value for fsetpos other than a value returned from fgetpos	No
TS17961:5.41-libuse	5.41. Using an object overwritten by getenv, localeconv, setlocale, and strerror	No
TS17961:5.42-chreof	5.42. Using character values that are indistinguishable from EOF	No
TS17961:5.43-resident	5.43. Using identifiers that are reserved for the implementation	No
TS17961:5.44-invfmtstr	5.44. Using invalid format strings	Yes
TS17961:5.45-taintsink	5.45. Tainted, potentially mutilated, or out-of-domain integer values are used in a restricted sink	Yes

GrammaTech is a leading global provider of application testing (AST) solutions used by the world's most security conscious organizations to detect, measure, analyze and resolve vulnerabilities for software they develop or use. The company is also a trusted cybersecurity and artificial intelligence research partner for the nation's civil, defense, and intelligence agencies.

CodeSonar and CodeSentry are registered trademarks of GrammaTech, Inc.
© GrammaTech, Inc. All rights reserved.

