Klocwork

This page was automatically generated and should not be edited.

The information on this page was provided by outside contributors and has not been verified by SEI CERT.

The table below can be re-ordered, by clicking column headers.

Tool Version: 2018

Checker	Guideline
ABV.ANY_SIZE_ARRAY	ARR30-C. Do not form or use out-of-bounds pointers or array subscripts
ABV.ANY_SIZE_ARRAY	ARR38-C. Guarantee that library functions do not form invalid pointers
ABV.ANY_SIZE_ARRAY	ARR00-C. Understand how arrays work
ABV.ANY_SIZE_ARRAY	ENV01-C. Do not make assumptions about the size of an environment variable
ABV.GENERAL	ARR30-C. Do not form or use out-of-bounds pointers or array subscripts
ABV.GENERAL	ARR38-C. Guarantee that library functions do not form invalid pointers
ABV.GENERAL	ARR00-C. Understand how arrays work
ABV.GENERAL	ENV01-C. Do not make assumptions about the size of an environment variable
ABV.GENERAL	EXP08-C. Ensure pointer arithmetic is used correctly
ABV.ITERATOR	ARR38-C. Guarantee that library functions do not form invalid pointers
ABV.ITERATOR	ARR00-C. Understand how arrays work
ABV.ITERATOR	ENV01-C. Do not make assumptions about the size of an environment variable
ABV.ITERATOR	EXP08-C. Ensure pointer arithmetic is used correctly
ABV.MEMBER	ARR00-C. Understand how arrays work
ABV.MEMBER	ENV01-C. Do not make assumptions about the size of an environment variable
ABV.STACK	ARR30-C. Do not form or use out-of-bounds pointers or array subscripts
ABV.STACK	ARR38-C. Guarantee that library functions do not form invalid pointers
ABV.STACK	ARR00-C. Understand how arrays work
ABV.STACK	ENV01-C. Do not make assumptions about the size of an environment variable
ABV.TAINTED	ARR30-C. Do not form or use out-of-bounds pointers or array subscripts
ABV.TAINTED	ARR38-C. Guarantee that library functions do not form invalid pointers
ABV.TAINTED	ARR00-C. Understand how arrays work
ABV.TAINTED	ENV01-C. Do not make assumptions about the size of an environment variable
ABV.UNICODE.BOUND_MAP	ARR30-C. Do not form or use out-of-bounds pointers or array subscripts
ABV.UNICODE.BOUND_MAP	ARR00-C. Understand how arrays work
ABV.UNICODE.BOUND_MAP	ENV01-C. Do not make assumptions about the size of an environment variable
ABV.UNICODE.FAILED_MAP	ARR30-C. Do not form or use out-of-bounds pointers or array subscripts
ABV.UNICODE.FAILED_MAP	ARR00-C. Understand how arrays work
ABV.UNICODE.FAILED_MAP	ENV01-C. Do not make assumptions about the size of an environment variable
ABV.UNICODE.NNTS_MAP	ARR30-C. Do not form or use out-of-bounds pointers or array subscripts
ABV.UNICODE.NNTS_MAP	ARR00-C. Understand how arrays work
ABV.UNICODE.NNTS_MAP	ENV01-C. Do not make assumptions about the size of an environment variable
ABV.UNICODE.SELF_MAP	ARR30-C. Do not form or use out-of-bounds pointers or array subscripts
ABV.UNICODE.SELF_MAP	ARR00-C. Understand how arrays work
ABV.UNICODE.SELF_MAP	ENV01-C. Do not make assumptions about the size of an environment variable

ABV.UNKNOWN_SIZE	ARR30-C. Do not form or use out-of-bounds pointers or array subscripts
ABV.UNKNOWN_SIZE	ARR38-C. Guarantee that library functions do not form invalid pointers
ABV.UNKNOWN_SIZE	ARR00-C. Understand how arrays work
ABV.UNKNOWN_SIZE	ENV01-C. Do not make assumptions about the size of an environment variable
ASSIGCOND.CALL	EXP45-C. Do not perform assignments in selection statements
ASSIGCOND.GEN	EXP45-C. Do not perform assignments in selection statements
BYTEORDER.HTON.SEND	POS39-C. Use the correct byte ordering when transferring data between systems
BYTEORDER.HTON.WRITE	POS39-C. Use the correct byte ordering when transferring data between systems
BYTEORDER.NTOH.READ	POS39-C. Use the correct byte ordering when transferring data between systems
BYTEORDER.NTOH.RECV	POS39-C. Use the correct byte ordering when transferring data between systems
CONC.DL	CON35-C. Avoid deadlock by locking in a predefined order
CONC.DL	POS51-C. Avoid deadlock with POSIX threads by locking in predefined order
CONC.SLEEP	POS52-C. Do not perform operations that can block while holding a POSIX lock
CONC.SLEEP	CON05-C. Do not perform operations that can block while holding a lock
CWARN.CMPCHR.EOF	FIO34-C. Distinguish between characters read from a file and EOF or WEOF
CWARN.EMPTY.LABEL	MSC01-C. Strive for logical completeness
CWARN.FUNCADDR	EXP16-C. Do not compare function pointers to constant values
CWARN.IMPLICITINT	DCL31-C. Declare identifiers before using them
CWARN.MEMSET.SIZEOF.PTR	ARR01-C. Do not apply the sizeof operator to a pointer when taking the size of an array
CWARN.NOEFFECT.OUTOFRANGE	INT30-C. Ensure that unsigned integer operations do not wrap
CWARN.NOEFFECT.SELF_ASSIGN	MSC12-C. Detect and remove code that has no effect or is never executed
CWARN.NOEFFECT.UCMP.GE	MSC12-C. Detect and remove code that has no effect or is never executed
CWARN.NOEFFECT.UCMP.GE.MACRO	MSC12-C. Detect and remove code that has no effect or is never executed
CWARN.NOEFFECT.UCMP.LT	MSC12-C. Detect and remove code that has no effect or is never executed
CWARN.NOEFFECT.UCMP.LT.MACRO	MSC12-C. Detect and remove code that has no effect or is never executed
CWARN.NULLCHECK.FUNCNAME	EXP16-C. Do not compare function pointers to constant values
CWARN.NULLCHECK.FUNCNAME	MSC12-C. Detect and remove code that has no effect or is never executed
DBZ.CONST	INT33-C. Ensure that division and remainder operations do not result in divide-by-zero errors
DBZ.CONST.CALL	INT33-C. Ensure that division and remainder operations do not result in divide-by-zero errors
DBZ.GENERAL	INT33-C. Ensure that division and remainder operations do not result in divide-by-zero errors
DBZ.ITERATOR	INT33-C. Ensure that division and remainder operations do not result in divide-by-zero errors
EFFECT	MSC12-C. Detect and remove code that has no effect or is never executed
FMM.MIGHT	WIN30-C. Properly pair allocation and deallocation functions
FMM.MUST	WIN30-C. Properly pair allocation and deallocation functions
FNH.MIGHT	MEM34-C. Only free memory allocated dynamically
FNH.MIGHT	MEM00-C. Allocate and free memory in the same module, at the same level of abstraction
FNH.MUST	MEM34-C. Only free memory allocated dynamically
FNH.MUST	MEM00-C. Allocate and free memory in the same module, at the same level of abstraction
FREE.INCONSISTENT	MEM00-C. Allocate and free memory in the same module, at the same level of abstraction
FUM.GEN.MIGHT	MEM34-C. Only free memory allocated dynamically
FUM.GEN.MIGHT	MEM00-C. Allocate and free memory in the same module, at the same level of abstraction
FUM.GEN.MUST	MEM34-C. Only free memory allocated dynamically
FUM.GEN.MUST	MEM00-C. Allocate and free memory in the same module, at the same level of abstraction
FUNCRET.GEN	MSC37-C. Ensure that control never reaches the end of a non-void function
FUNCRET.IMPLICIT	DCL31-C. Declare identifiers before using them
INCORRECT.ALLOC_SIZE	MEM35-C. Allocate sufficient memory for an object
INFINITE_LOOP.GLOBAL	MSC01-C. Strive for logical completeness

INFINITE_LOOP.LOCAL	MSC01-C. Strive for logical completeness
INFINITE_LOOP.MACRO	MSC01-C. Strive for logical completeness
INVARIANT_CONDITION.UNREACH	MSC07-C. Detect and remove dead code
INVARIANT_CONDITION.UNREACH	MSC12-C. Detect and remove code that has no effect or is never executed
LA_UNUSED	MSC01-C. Strive for logical completeness
LA_UNUSED	MSC07-C. Detect and remove dead code
LA_UNUSED	MSC12-C. Detect and remove code that has no effect or is never executed
LOCRET.ARG	DCL30-C. Declare objects with appropriate storage durations
LOCRET.GLOB	DCL30-C. Declare objects with appropriate storage durations
LOCRET.RET	DCL30-C. Declare objects with appropriate storage durations
LV_UNUSED.GEN	MSC13-C. Detect and remove unused values
MISRA.ASSIGN.COND	EXP45-C. Do not perform assignments in selection statements
MISRA.BITFIELD.TYPE	INT12-C. Do not make assumptions about the type of a plain int bit-field when used in an expression
MISRA.BITS.NOT_UNSIGNED	INT13-C. Use bitwise operators only on unsigned operands
MISRA.BITS.NOT_UNSIGNED.PREP	INT13-C. Use bitwise operators only on unsigned operands
MISRA.CAST.FUNC_PTR.2012	DCL07-C. Include the appropriate type information in function declarators
MISRA.CAST.INT	INT02-C. Understand integer conversion rules
MISRA.CAST.OBJ_PTR_TO_INT.2012	INT36-C. Converting a pointer to integer or integer to pointer
MISRA.CAST.PTR.UNRELATED	EXP36-C. Do not cast pointers into more strictly aligned pointer types
MISRA.CAST.PTR_TO_INT	EXP36-C. Do not cast pointers into more strictly aligned pointer types
MISRA.CAST.UNSIGNED_BITS	INT02-C. Understand integer conversion rules
MISRA.CONV.INT.SIGN	INT02-C. Understand integer conversion rules
MISRA.CVALUE.IMPL.CAST	INT02-C. Understand integer conversion rules
MISRA.DECL.NO_TYPE	DCL31-C. Declare identifiers before using them
MISRA.DEFINE.BADEXP	PRE02-C. Macro replacement lists should be parenthesized
MISRA.DEFINE.BADEXP	PRE10-C. Wrap multistatement macros in a do-while loop
MISRA.DEFINE.FUNC	PRE00-C. Prefer inline or static functions to function-like macros
MISRA.DEFINE.NOPARS	PRE01-C. Use parentheses within macros around parameter names
MISRA.DEFINE.SHARP.ORDER.2012	PRE05-C. Understand macro replacement when concatenating tokens or performing stringification
MISRA.DEFINE.WRONGNAME.UNDERSCORE	DCL37-C. Do not declare or define a reserved identifier
MISRA.ENUM.IMPLICIT.VAL.NON_UNIQUE.2012	INT09-C. Ensure enumeration constants map to unique values
MISRA.EXPR.PARENS.2012	EXP00-C. Use parentheses for precedence of operation
MISRA.FOR.COND.FLT	FLP30-C. Do not use floating-point variables as loop counters
MISRA.FOR.COUNTER.FLT	FLP30-C. Do not use floating-point variables as loop counters
MISRA.FUNC.NO_PARAMS	DCL20-C. Explicitly specify void when a function accepts no arguments
MISRA.FUNC.NOPROT.CALL	DCL31-C. Declare identifiers before using them
MISRA.FUNC.NOPROT.DEF	DCL07-C. Include the appropriate type information in function declarators
MISRA.FUNC.PROT_FORM.KR.2012	DCL07-C. Include the appropriate type information in function declarators
MISRA.FUNC.RECUR	MEM05-C. Avoid large stack allocations
MISRA.FUNC.STATIC.REDECL	DCL36-C. Do not declare an identifier with conflicting linkage classifications
MISRA.FUNC.UNMATCHED.PARAMS	EXP37-C. Call functions with the correct number and type of arguments
MISRA.FUNC.UNUSEDRET.2012	EXP12-C. Do not ignore values returned by functions
MISRA.FUNC.VARARG	DCL11-C. Understand the type issues associated with variadic functions
MISRA.IDENT.DISTINCT.C99.2012	DCL23-C. Guarantee that mutually visible identifiers are unique
MISRA.IF.NO_COMPOUND	EXP19-C. Use braces for the body of an if, for, or while statement
MISRA.IF.NO_ELSE	MSC01-C. Strive for logical completeness
MISRA.INCGUARD	PRE06-C. Enclose header files in an include guard

MISRA.LOGIC.OPERATOR.NOT_BOOL	EXP46-C. Do not use a bitwise operator with a Boolean-like operand
MISRA.LOGIC.SIDEEFF	EXP02-C. Be aware of the short-circuit behavior of the logical AND and OR operators
MISRA.PPARAM.NEEDS.CONST	DCL13-C. Declare function parameters that are pointers to values not changed by the function as const
MISRA.RETURN.NOT_LAST	MSC37-C. Ensure that control never reaches the end of a non-void function
MISRA.SIZEOF.SIDE_EFFECT	EXP44-C. Do not rely on side effects in operands to sizeof, _Alignof, or _Generic
MISRA.STDLIB.ABORT	ENV33-C. Do not call system()
MISRA.STDLIB.ATOI	ERR34-C. Detect errors when converting a string to a number
MISRA.STDLIB.WRONGNAME	DCL37-C. Do not declare or define a reserved identifier
MISRA.STDLIB.WRONGNAME.UNDERSCORE	DCL37-C. Do not declare or define a reserved identifier
MISRA.STMT.NO_COMPOUND	EXP19-C. Use braces for the body of an if, for, or while statement
MISRA.STMT.NO_EFFECT	MSC12-C. Detect and remove code that has no effect or is never executed
MISRA.STRING_LITERAL.NON_CONST.2012	STR05-C. Use pointers to const when referring to string literals
MISRA.SWITCH.WELL_FORMED.BREAK.2012	MSC17-C. Finish every set of statements associated with a case label with a break statement
MISRA.SWITCH.WELL_FORMED.DEFAULT.2012	MSC01-C. Strive for logical completeness
MISRA.TOKEN.OCTAL.ESCAPE	DCL18-C. Do not begin integer constants with 0 when specifying a decimal value
MISRA.TOKEN.OCTAL.INT	DCL18-C. Do not begin integer constants with 0 when specifying a decimal value
MISRA.UMINUS.UNSIGNED	INT02-C. Understand integer conversion rules
MISRA.VAR.HIDDEN	DCL01-C. Do not reuse variable names in subscopes
MLK.MIGHT	MEM31-C. Free dynamically allocated memory when no longer needed
MLK.MIGHT	MEM00-C. Allocate and free memory in the same module, at the same level of abstraction
MLK.MIGHT	MEM12-C. Consider using a goto chain when leaving a function on error when using and releasing resources
MLK.MUST	MEM31-C. Free dynamically allocated memory when no longer needed
MLK.MUST	MEM00-C. Allocate and free memory in the same module, at the same level of abstraction
MLK.MUST	MEM12-C. Consider using a goto chain when leaving a function on error when using and releasing resources
MLK.RET.MIGHT	MEM31-C. Free dynamically allocated memory when no longer needed
MLK.RET.MIGHT	MEM00-C. Allocate and free memory in the same module, at the same level of abstraction
MLK.RET.MIGHT	MEM12-C. Consider using a goto chain when leaving a function on error when using and releasing resources
MLK.RET.MUST	MEM31-C. Free dynamically allocated memory when no longer needed
MLK.RET.MUST	MEM00-C. Allocate and free memory in the same module, at the same level of abstraction
MLK.RET.MUST	MEM12-C. Consider using a goto chain when leaving a function on error when using and releasing resources
NNTS.MIGHT	ARR30-C. Do not form or use out-of-bounds pointers or array subscripts
NNTS.MIGHT	STR31-C. Guarantee that storage for strings has sufficient space for character data and the null terminator
NNTS.MIGHT	STR32-C. Do not pass a non-null-terminated character sequence to a library function that expects a string
NNTS.MIGHT	ARR00-C. Understand how arrays work
NNTS.MIGHT	STR03-C. Do not inadvertently truncate a string
NNTS.MUST	ARR30-C. Do not form or use out-of-bounds pointers or array subscripts
NNTS.MUST	STR31-C. Guarantee that storage for strings has sufficient space for character data and the null terminator
NNTS.MUST	STR32-C. Do not pass a non-null-terminated character sequence to a library function that expects a string
NNTS.MUST	ARR00-C. Understand how arrays work
NNTS.MUST	STR03-C. Do not inadvertently truncate a string
NNTS.TAINTED	ARR30-C. Do not form or use out-of-bounds pointers or array subscripts
NNTS.TAINTED	STR32-C. Do not pass a non-null-terminated character sequence to a library function that expects a string
NNTS.TAINTED	ARR00-C. Understand how arrays work
NNTS.TAINTED	STR02-C. Sanitize data passed to complex subsystems
NPD.CHECK.CALL.MIGHT	EXP34-C. Do not dereference null pointers

NPD.CHECK.CALL.MUST	EXP34-C. Do not dereference null pointers
NPD.CHECK.MIGHT	EXP34-C. Do not dereference null pointers
NPD.CHECK.MUST	EXP34-C. Do not dereference null pointers
NPD.CONST.CALL	EXP34-C. Do not dereference null pointers
NPD.CONST.DEREF	EXP34-C. Do not dereference null pointers
NPD.FUNC.CALL.MIGHT	EXP34-C. Do not dereference null pointers
NPD.FUNC.CALL.MUST	EXP34-C. Do not dereference null pointers
NPD.FUNC.MIGHT	EXP34-C. Do not dereference null pointers
NPD.FUNC.MUST	EXP34-C. Do not dereference null pointers
NPD.GEN.CALL.MIGHT	EXP34-C. Do not dereference null pointers
NPD.GEN.CALL.MUST	EXP34-C. Do not dereference null pointers
NPD.GEN.MIGHT	EXP34-C. Do not dereference null pointers
NPD.GEN.MUST	EXP34-C. Do not dereference null pointers
NUM.OVERFLOW	INT30-C. Ensure that unsigned integer operations do not wrap
PORTING.CAST.PTR	EXP36-C. Do not cast pointers into more strictly aligned pointer types
PORTING.CAST.PTR.FLTPNT	EXP36-C. Do not cast pointers into more strictly aligned pointer types
PORTING.CAST.PTR.SIZE	EXP36-C. Do not cast pointers into more strictly aligned pointer types
PORTING.CAST.SIZE	EXP36-C. Do not cast pointers into more strictly aligned pointer types
PORTING.SIGNED.CHAR	INT07-C. Use only explicitly signed or unsigned char type for numeric values
PORTING.STORAGE.STRUCT	DCL39-C. Avoid information leakage when passing a structure across a trust boundary
PORTING.STRUCT.BOOL	DCL39-C. Avoid information leakage when passing a structure across a trust boundary
PORTING.VAR.EFFECTS	EXP30-C. Do not depend on the order of evaluation for side effects
PRECISION.LOSS	INT31-C. Ensure that integer conversions do not result in lost or misinterpreted data
PRECISION.LOSS	INT02-C. Understand integer conversion rules
PRECISION.LOSS.CALL	INT31-C. Ensure that integer conversions do not result in lost or misinterpreted data
RETVOID.IMPLICIT	DCL31-C. Declare identifiers before using them
RH.LEAK	FIO42-C. Close files when they are no longer needed
RH.LEAK	FIO22-C. Close files before spawning processes
RH.LEAK	MEM00-C. Allocate and free memory in the same module, at the same level of abstraction
RH.LEAK	MEM12-C. Consider using a goto chain when leaving a function on error when using and releasing resources
RNPD.CALL	EXP34-C. Do not dereference null pointers
RNPD.DEREF	EXP34-C. Do not dereference null pointers
SEMICOL	EXP15-C. Do not place a semicolon on the same line as an if, for, or while statement
SV.BRM.HKEY_LOCAL_MACHINE	POS02-C. Follow the principle of least privilege
SV.CODE_INJECTION.SHELL_EXEC	ENV33-C. Do not call system()
SV.DLLPRELOAD.NONABSOLUTE.DLL	FIO02-C. Canonicalize path names originating from tainted sources
SV.DLLPRELOAD.NONABSOLUTE.DLL	WIN00-C. Be specific when dynamically loading libraries
SV.DLLPRELOAD.NONABSOLUTE.EXE	WIN00-C. Be specific when dynamically loading libraries
SV.DLLPRELOAD.SEARCHPATH	WIN00-C. Be specific when dynamically loading libraries
SV.FIU.PROCESS_VARIANTS	POS36-C. Observe correct revocation order while relinquishing privileges
SV.FIU.PROCESS_VARIANTS	POS37-C. Ensure that privilege relinquishment is successful
SV.FMT_STR.PRINT_FORMAT_MISMATCH.BAD	FIO47-C. Use valid format strings
SV.FMT_STR.PRINT_FORMAT_MISMATCH.BAD	DCL11-C. Understand the type issues associated with variadic functions
SV.FMT_STR.PRINT_FORMAT_MISMATCH.UNDESIRED	FIO47-C. Use valid format strings
SV.FMT_STR.PRINT_FORMAT_MISMATCH.UNDESIRED	DCL11-C. Understand the type issues associated with variadic functions

SV.FMT_STR.PRINT_IMPROP_LENGTH	FIO47-C. Use valid format strings
SV.FMT_STR.PRINT_IMPROP_LENGTH	DCL11-C. Understand the type issues associated with variadic functions
SV.FMT_STR.PRINT_PARAMS_WRONGNUM.FEW	FIO47-C. Use valid format strings
SV.FMT_STR.PRINT_PARAMS_WRONGNUM.FEW	DCL10-C. Maintain the contract between the writer and caller of variadic functions
SV.FMT_STR.PRINT_PARAMS_WRONGNUM.FEW	DCL11-C. Understand the type issues associated with variadic functions
SV.FMT_STR.PRINT_PARAMS_WRONGNUM.MANY	FIO47-C. Use valid format strings
SV.FMT_STR.PRINT_PARAMS_WRONGNUM.MANY	DCL10-C. Maintain the contract between the writer and caller of variadic functions
SV.FMT_STR.PRINT_PARAMS_WRONGNUM.MANY	DCL11-C. Understand the type issues associated with variadic functions
SV.FMT_STR.SCAN_FORMAT_MISMATCH.BAD	FIO47-C. Use valid format strings
SV.FMT_STR.SCAN_FORMAT_MISMATCH.BAD	DCL11-C. Understand the type issues associated with variadic functions
SV.FMT_STR.SCAN_FORMAT_MISMATCH.UNDESIRED	FIO47-C. Use valid format strings
SV.FMT_STR.SCAN_FORMAT_MISMATCH.UNDESIRED	DCL11-C. Understand the type issues associated with variadic functions
SV.FMT_STR.SCAN_IMPROP_LENGTH	FIO47-C. Use valid format strings
SV.FMT_STR.SCAN_PARAMS_WRONGNUM.FEW	FIO47-C. Use valid format strings
SV.FMT_STR.SCAN_PARAMS_WRONGNUM.FEW	DCL10-C. Maintain the contract between the writer and caller of variadic functions
SV.FMT_STR.SCAN_PARAMS_WRONGNUM.MANY	FIO47-C. Use valid format strings
SV.FMT_STR.SCAN_PARAMS_WRONGNUM.MANY	DCL10-C. Maintain the contract between the writer and caller of variadic functions
SV.FMT_STR.UNKWN_FORMAT	FIO47-C. Use valid format strings
SV.FMT_STR.UNKWN_FORMAT.SCAN	DCL11-C. Understand the type issues associated with variadic functions
SV.FMTSTR.GENERIC	FIO30-C. Exclude user input from format strings
SV.INCORRECT_RESOURCE_HANDLING.URH	FIO46-C. Do not access a closed file
SV.RVT.RETVAL_NOTTESTED	POS54-C. Detect and handle POSIX library errors
SV.RVT.RETVAL_NOTTESTED	EXP12-C. Do not ignore values returned by functions
SV.STRBO.BOUND_COPY.OVERFLOW	ARR30-C. Do not form or use out-of-bounds pointers or array subscripts
SV.STRBO.BOUND_COPY.OVERFLOW	STR31-C. Guarantee that storage for strings has sufficient space for character data and the null terminator
SV.STRBO.BOUND_COPY.OVERFLOW	ARR00-C. Understand how arrays work
SV.STRBO.BOUND_COPY.UNTERM	ARR30-C. Do not form or use out-of-bounds pointers or array subscripts
SV.STRBO.BOUND_COPY.UNTERM	STR31-C. Guarantee that storage for strings has sufficient space for character data and the null terminator
SV.STRBO.BOUND_COPY.UNTERM	ARR00-C. Understand how arrays work
SV.STRBO.BOUND_SPRINTF	ARR30-C. Do not form or use out-of-bounds pointers or array subscripts
SV.STRBO.BOUND_SPRINTF	STR31-C. Guarantee that storage for strings has sufficient space for character data and the null terminator
SV.STRBO.BOUND_SPRINTF	ARR00-C. Understand how arrays work
SV.STRBO.UNBOUND_COPY	STR31-C. Guarantee that storage for strings has sufficient space for character data and the null terminator
SV.STRBO.UNBOUND_COPY	ARR00-C. Understand how arrays work
SV.STRBO.UNBOUND_SPRINTF	STR31-C. Guarantee that storage for strings has sufficient space for character data and the null terminator
SV.STRBO.UNBOUND_SPRINTF	ARR00-C. Understand how arrays work
SV.TAINTED.ALLOC_SIZE	ARR30-C. Do not form or use out-of-bounds pointers or array subscripts
SV.TAINTED.ALLOC_SIZE	ARR00-C. Understand how arrays work
SV.TAINTED.ALLOC_SIZE	INT04-C. Enforce limits on integer values originating from tainted sources
SV.TAINTED.BINOP	INT04-C. Enforce limits on integer values originating from tainted sources
SV.TAINTED.CALL.BINOP	INT04-C. Enforce limits on integer values originating from tainted sources
SV.TAINTED.CALL.INDEX_ACCESS	ARR30-C. Do not form or use out-of-bounds pointers or array subscripts
SV.TAINTED.CALL.INDEX_ACCESS	ARR00-C. Understand how arrays work
SV.TAINTED.CALL.INDEX_ACCESS	INT04-C. Enforce limits on integer values originating from tainted sources
SV.TAINTED.CALL.LOOP_BOUND	ARR30-C. Do not form or use out-of-bounds pointers or array subscripts
SV.TAINTED.CALL.LOOP_BOUND	ARR00-C. Understand how arrays work

SV.TAINTED.CALL.LOOP_BOUND	INT04-C. Enforce limits on integer values originating from tainted sources
SV.TAINTED.FMTSTR	FIO30-C. Exclude user input from format strings
SV.TAINTED.INDEX_ACCESS	ARR30-C. Do not form or use out-of-bounds pointers or array subscripts
SV.TAINTED.INDEX_ACCESS	ARR00-C. Understand how arrays work
SV.TAINTED.INDEX_ACCESS	INT04-C. Enforce limits on integer values originating from tainted sources
SV.TAINTED.INJECTION	ENV33-C. Do not call system()
SV.TAINTED.INJECTION	STR02-C. Sanitize data passed to complex subsystems
SV.TAINTED.LOOP_BOUND	ARR30-C. Do not form or use out-of-bounds pointers or array subscripts
SV.TAINTED.LOOP_BOUND	ARR00-C. Understand how arrays work
SV.TAINTED.LOOP_BOUND	INT04-C. Enforce limits on integer values originating from tainted sources
SV.TOCTOU.FILE_ACCESS	FIO45-C. Avoid TOCTOU race conditions while accessing files
SV.TOCTOU.FILE_ACCESS	POS35-C. Avoid race conditions while checking for the existence of a symbolic link
SV.TOCTOU.FILE_ACCESS	FIO01-C. Be careful using functions that use file names for identification
SV.TOCTOU.FILE_ACCESS	FIO02-C. Canonicalize path names originating from tainted sources
SV.UNBOUND_STRING_INPUT.CIN	ARR30-C. Do not form or use out-of-bounds pointers or array subscripts
SV.UNBOUND_STRING_INPUT.CIN	ARR00-C. Understand how arrays work
SV.UNBOUND_STRING_INPUT.FUNC	ARR30-C. Do not form or use out-of-bounds pointers or array subscripts
SV.UNBOUND_STRING_INPUT.FUNC	ARR00-C. Understand how arrays work
SV.USAGERULES.PERMISSIONS	POS36-C. Observe correct revocation order while relinquishing privileges
SV.USAGERULES.PERMISSIONS	POS37-C. Ensure that privilege relinquishment is successful
SV.USAGERULES.PERMISSIONS	POS02-C. Follow the principle of least privilege
SV.USAGERULES.PROCESS_VARIANTS	POS36-C. Observe correct revocation order while relinquishing privileges
UFM.DEREF.MIGHT	MEM30-C. Do not access freed memory
UFM.DEREF.MIGHT	MEM00-C. Allocate and free memory in the same module, at the same level of abstraction
UFM.DEREF.MUST	MEM30-C. Do not access freed memory
UFM.DEREF.MUST	MEM00-C. Allocate and free memory in the same module, at the same level of abstraction
UFM.FFM.MIGHT	MEM30-C. Do not access freed memory
UFM.FFM.MIGHT	MEM00-C. Allocate and free memory in the same module, at the same level of abstraction
UFM.FFM.MUST	MEM30-C. Do not access freed memory
UFM.FFM.MUST	MEM00-C. Allocate and free memory in the same module, at the same level of abstraction
UFM.RETURN.MIGHT	MEM30-C. Do not access freed memory
UFM.RETURN.MIGHT	MEM00-C. Allocate and free memory in the same module, at the same level of abstraction
UFM.RETURN.MUST	MEM30-C. Do not access freed memory
UFM.RETURN.MUST	MEM00-C. Allocate and free memory in the same module, at the same level of abstraction
UFM.USE.MIGHT	MEM30-C. Do not access freed memory
UFM.USE.MIGHT	MEM00-C. Allocate and free memory in the same module, at the same level of abstraction
UFM.USE.MUST	MEM30-C. Do not access freed memory
UFM.USE.MUST	MEM00-C. Allocate and free memory in the same module, at the same level of abstraction
UNINIT.HEAP.MIGHT	EXP33-C. Do not read uninitialized memory
UNINIT.HEAP.MUST	EXP33-C. Do not read uninitialized memory
UNINIT.STACK.ARRAY.MIGHT	EXP33-C. Do not read uninitialized memory
UNINIT.STACK.ARRAY.MUST	EXP33-C. Do not read uninitialized memory
UNINIT.STACK.ARRAY.PARTIAL.MUST	EXP33-C. Do not read uninitialized memory
UNINIT.STACK.MIGHT	EXP33-C. Do not read uninitialized memory
UNINIT.STACK.MUST	EXP33-C. Do not read uninitialized memory
UNREACH.GEN	MSC07-C. Detect and remove dead code
UNREACH.GEN	MSC12-C. Detect and remove code that has no effect or is never executed

UNREACH.RETURN	MSC07-C. Detect and remove dead code
UNREACH.RETURN	MSC12-C. Detect and remove code that has no effect or is never executed
UNREACH.SIZEOF	MSC07-C. Detect and remove dead code
UNREACH.SIZEOF	MSC12-C. Detect and remove code that has no effect or is never executed
VA_UNUSED.GEN	MSC12-C. Detect and remove code that has no effect or is never executed
VA_UNUSED.GEN	MSC13-C. Detect and remove unused values
VA_UNUSED.INIT	MSC12-C. Detect and remove code that has no effect or is never executed
VA_UNUSED.INIT	MSC13-C. Detect and remove unused values