

Axivion Bauhaus Suite – Technical Factsheet CERT

Version 6.9 upwards

Contents

1. C	2
1. Cert C 2016.....	2
2. C++	8
2. Cert C++ 2016	8

1. C

1. Cert C 2016

CERT Rule	Severity	Description
PRE00	Medium	Prefer inline or static functions to function-like macros.
PRE01	Medium	Use parentheses within macros around parameter names.
PRE02	Medium	Macro replacement lists should be parenthesized.
PRE03	Low	Prefer typedefs to defines for encoding non-pointer types.
PRE04	Low	Do not reuse a standard header file name.
PRE05	Low	Understand macro replacement when concatenating tokens or performing stringification.
PRE06	Low	Enclose header files in an inclusion guard.
PRE07	Low	Avoid using repeated question marks.
PRE08	Low	Guarantee that header file names are unique.
PRE09	High	Do not replace secure functions with deprecated or obsolescent functions.
PRE10	Medium	Wrap multistatement macros in a do-while loop.
PRE11	Medium	Do not conclude macro definitions with a semicolon.
PRE12	Low	Do not define unsafe macros.
PRE13	Low	Use the Standard predefined macros to test for versions and features.
PRE30	Low	Do not create a universal character name through concatenation.
PRE31	Low	Avoid side effects in arguments to unsafe macros.
PRE32	Low	Do not use preprocessor directives in invocations of function-like macros.
DCL00	Low	Const-qualify immutable objects.
DCL01	Low	Do not reuse variable names in subscopes.
DCL02	Low	Use visually distinct identifiers.

DCL03	Low	Use a static assertion to test the value of a constant expression.
DCL04	Low	Do not declare more than one variable per declaration.
DCL05	Low	Use typedefs of non-pointer types only.
DCL06	Low	Use meaningful symbolic constants to represent literal values.
DCL07	Low	Include the appropriate type information in function declarators.
DCL09	Low	Declare functions that return errno with a return type of errno_t.
DCL11	High	Understand the type issues associated with variadic functions.
DCL12	Low	Implement abstract data types using opaque types.
DCL13	Low	Declare function parameters that are pointers to values not changed by the function as const.
DCL15	Low	Declare file-scope objects or functions that do not need external linkage as static.
DCL16	Low	Use "L", not "l", to indicate a long value.
DCL18	Low	Do not begin integer constants with 0 when specifying a decimal value.
DCL19	Low	Minimize the scope of variables and functions.
DCL20	Medium	Explicitly specify void when a function accepts no arguments.
DCL21	Low	Understand the storage of compound literals.
DCL23	Medium	Guarantee that mutually visible identifiers are unique.
DCL30	High	Declare objects with appropriate storage durations.
DCL31	Low	Declare identifiers before using them.
DCL36	Medium	Do not declare an identifier with conflicting linkage classifications.
DCL37	Low	Do not declare or define a reserved identifier.
DCL38	Low	Use the correct syntax when declaring a flexible array member.
DCL39	Low	Avoid information leakage when passing a structure across a trust boundary.
DCL40	Low	Do not create incompatible declarations of the same function or object.

DCL41	Medium	Do not declare variables inside a switch statement before the first case label.
EXP00	Low	Use parentheses for precedence of operation.
EXP02	Low	Be aware of the short-circuit behaviour of the logical AND and OR operators.
EXP05	Medium	Do not cast away a const qualification.
EXP07	Low	Do not diminish the benefits of constants by assuming their values in expressions.
EXP10	Medium	Do not depend on the order of evaluation of subexpressions or the order in which side effects take place.
EXP12	Medium	Do not ignore values returned by functions.
EXP14	Low	Beware of integer promotion when performing bitwise operations on integer types smaller than int.
EXP15	High	Do not place a semicolon on the same line as an if, for, or while statement.
EXP19	Medium	Use braces for the body of an if, for, or while statement.
EXP20	Medium	Perform explicit tests to determine success, true and false, and equality.
EXP30	Medium	Do not depend on the order of evaluation for side effects.
EXP32	Low	Do not access a volatile object through a nonvolatile reference.
EXP33	High	Do not read uninitialized memory.
EXP34	High	Do not dereference null pointers
EXP35	Low	Do not modify objects with temporary lifetime.
EXP36	Low	Do not cast pointers into more strictly aligned pointer types.
EXP37	Medium	Call functions with the correct number and type of arguments.
EXP40	Low	Do not modify constant objects.
EXP42	Medium	Do not compare padding data.
EXP44	Low	Do not rely on side effects in operands to sizeof, _Alignof, or _Generic.

EXP45	Low	Do not perform assignments in selection statements.
EXP46	Low	Do not use a bitwise operator with a Boolean-like operand.
EXP47	Medium	Do not call <code>va_arg</code> with an argument of the incorrect type.
INT00	High	Understand the data model used by your implementation(s).
INT01	Medium	Use <code>rsize_t</code> or <code>size_t</code> for all integer values representing the size of an object.
INT05	Medium	Do not use input functions to convert character data if they cannot handle all possible inputs.
INT07	Medium	Use only explicitly signed or unsigned <code>char</code> type for numeric values.
INT08	Medium	Verify that all integer values are in range.
INT09	Low	Ensure enumeration constants map to unique values.
INT12	Low	Do not make assumptions about the type of a plain int bit-field when used in an expression.
INT13	High	Use bitwise operators only on unsigned operands.
INT15	High	Use <code>intmax_t</code> or <code>uintmax_t</code> for formatted IO on programmer-defined integer types.
INT17	High	Define integer constants in an implementation-independent manner.
INT34	Low	Do not shift an expression by a negative number of bits or by greater than or equal to the number of bits that exist in the operand.
INT36	Low	Converting a pointer to integer or integer to pointer.
FLP02	Low	Avoid using floating-point numbers when precise computation is needed.
FLP06	Low	Convert integers to floating point for floating-point operations.
FLP07	Low	Cast the return value of a function that returns a floating-point type.
FLP30	Low	Do not use floating-point variables as loop counters.
FLP32	Medium	Prevent or detect domain and range errors in math functions.
FLP37	Low	Do not use object representations to compare floating-point values.
ARR01	High	Do not apply the <code>sizeof</code> operator to a pointer when taking the size of an array.

ARR02	Medium	Explicitly specify array bounds, even if implicitly defined by an initializer.
ARR30	High	Do not form or use out-of-bounds pointers or array subscripts.
ARR36	Medium	Do not subtract or compare two pointers that do not refer to the same array.
ARR37	Medium	Do not add or subtract an integer to a pointer to a non-array object.
ARR39	High	Do not add or subtract a scaled integer to a pointer.
STR04	Low	Use plain char for characters in the basic character set.
STR05	Low	Use pointers to const when referring to string literals.
STR07	High	Use the bounds-checking interfaces for string manipulation.
STR09	Low	Don't assume numeric values for expressions with type plain character.
STR10	Low	Do not concatenate different type of string literals.
STR11	Low	Do not specify the bound of a character array initialized with a string literal.
STR30	Low	Do not attempt to modify string literals.
STR31	High	Guarantee that storage for strings has sufficient space for character data and the null terminator.
STR32	High	Do not pass a non-null-terminated character sequence to a library function that expects a string.
STR34	Medium	Cast characters to unsigned char before converting to larger integer sizes.
STR37	Low	Arguments to character-handling functions must be representable as an unsigned char.
STR38	High	Do not confuse narrow and wide character strings and functions.
MEM01	High	Store a new value in pointers immediately after free().
MEM02	Low	Immediately cast the result of a memory allocation function call into a pointer to the allocated type.
MEM30	High	Do not access freed memory.
MEM31	Medium	Free dynamically allocated memory when no longer needed.

MEM33	Low	Allocate and copy structures containing a flexible array member dynamically.
MEM34	High	Only free memory allocated dynamically.
MEM35	High	Allocate sufficient memory for an object.
MEM36	Low	Do not modify the alignment of objects by calling realloc().
ENV30	Low	Do not modify the object referenced by the return value of certain functions.
ENV32	Medium	All exit handlers must return normally.
ENV33	High	Do not call system().
FIO30	High	Exclude user input from format strings.
FIO34	High	Distinguish between characters read from a file and EOF or WEOF.
FIO37	High	Do not assume that fgets() or fgetws() returns a nonempty string when successful.
FIO38	Low	Do not copy a FILE object.
FIO47	High	Use valid format strings.
SIG30	High	Call only asynchronous-safe functions within signal handlers.
SIG31	High	Do not access shared objects in signal handlers.
SIG34	Low	Do not call signal() from within interruptible signal handlers.
SIG35	Low	Do not return from a computational exception signal handler.
ERR30	Medium	Set errno to zero before calling a library function known to set errno, and check errno only after the function returns a value indicating failure.
ERR32	Low	Do not rely on indeterminate values of errno.
ERR33	High	Detect and handle standard library errors.
ERR34	Medium	Detect errors when converting a string to a number.
CON32	Medium	Prevent data races when accessing bit-fields from multiple threads.
CON40	Medium	Do not refer to an atomic variable twice in an expression.
MSC24	High	Do not use deprecated or obsolescent functions.

MSC30	Medium	Do not use the rand() function for generating pseudorandom numbers.
MSC32	Medium	Properly seed pseudorandom number generators.
MSC33	High	Do not pass invalid data to the asctime() function.
MSC37	High	Ensure that control never reaches the end of a non-void function.
POS30	High	Use the readlink() function properly.
POS33	Low	Do not use vfork().
POS34	High	Do not call putenv() with a pointer to an automatic variable as the argument.
POS35	High	Avoid race conditions while checking for the existence of a symbolic link.
POS36	High	Observe correct revocation order while relinquishing privileges.
POS37	High	Ensure that privilege relinquishment is successful.
POS39	Medium	Use the correct byte ordering when transferring data between systems.
POS47	Medium	Do not use threads that can be canceled asynchronously.
POS49	Medium	When data must be accessed by multiple threads, provide a mutex and guarantee no adjacent data is also accessed.
POS54	High	Detect and handle POSIX library errors.

2. C++

2. Cert C++ 2016

CERT Rule	Severity	Description
DCL30	High	Declare objects with appropriate storage durations.
DCL39	Low	Avoid information leakage when passing a structure across a trust boundary.
DCL40	Low	Do not create incompatible declarations of the same function or object.
DCL50	High	Do not define a C-style variadic function.

DCL51	Low	Do not declare or define a reserved identifier.
DCL52	Low	Never qualify a reference type with const or volatile.
DCL55	Low	Avoid information leakage when passing a class object across a trust boundary.
DCL57	Low	Do not let exceptions escape from destructors or deallocation functions.
DCL58	High	Do not modify the standard namespaces.
DCL59	Medium	Do not define an unnamed namespace in a header file.
DCL60	High	Obey the one-definition rule.
EXP34	High	Do not dereference null pointers.
EXP35	Low	Do not modify objects with temporary lifetime.
EXP36	Low	Do not cast pointers into more strictly aligned pointer types.
EXP37	Medium	Call functions with the correct number and type of arguments.
EXP42	Medium	Do not compare padding data.
EXP45	Low	Do not perform assignments in selection statements.
EXP46	Low	Do not use a bitwise operator with a Boolean-like operand.
EXP47	Medium	Do not call va_arg with an argument of the incorrect type.
EXP50	Medium	Do not depend on the order of evaluation for side effects.
EXP52	Low	Do not rely on side effects in unevaluated operands.
EXP55	Medium	Do not access a cv-qualified object through a cv-unqualified type.
EXP59	Medium	Use offsetof() on valid types and members.
INT34	Low	Do not shift an expression by a negative number of bits or by greater than or equal to the number of bits that exist in the operand.
INT36	Low	Converting a pointer to integer or integer to pointer.
INT50	Medium	Do not cast to an out-of-range enumeration value.
ARR30	High	Do not form or use out-of-bounds pointers or array subscripts.
ARR37	Medium	Do not add or subtract an integer to a pointer to a non-array object.

ARR39	High	Do not add or subtract a scaled integer to a pointer.
CTR56	High	Do not use pointer arithmetic on polymorphic objects.
STR30	Low	Do not attempt to modify string literals.
STR31	High	Guarantee that storage for strings has sufficient space for character data and the null terminator.
STR32	High	Do not pass a non-null-terminated character sequence to a library function that expects a string.
STR34	Medium	Cast characters to unsigned char before converting to larger integer sizes.
STR37	Low	Arguments to character-handling functions must be representable as an unsigned char.
STR38	High	Do not confuse narrow and wide character strings and functions.
MEM30	High	Do not access freed memory.
MEM31	Medium	Free dynamically allocated memory when no longer needed.
MEM34	High	Only free memory allocated dynamically.
MEM35	High	Allocate sufficient memory for an object.
MEM36	Low	Do not modify the alignment of objects by calling realloc().
MEM50	High	Do not access freed memory.
MEM51	High	Properly deallocate dynamically allocated resources.
FIO30	High	Exclude user input from format strings.
FIO34	High	Distinguish between characters read from a file and EOF or WEOF.
FIO37	High	Do not assume that fgets() or fgetws() returns a nonempty string when successful.
FIO38	Low	Do not copy a FILE object.
FIO47	High	Use valid format strings.
ERR30	Medium	Set errno to zero before calling a library function known to set errno, and check errno only after the function returns a value indicating failure.
ERR32	Low	Do not rely on indeterminate values of errno.

ERR33	High	Detect and handle standard library errors.
ERR34	Medium	Detect errors when converting a string to a number.
ERR51	Low	Handle all exceptions.
ERR52	Low	Do not use <code>setjmp()</code> or <code>longjmp()</code> .
ERR53	Low	Do not reference base classes or class data members in a constructor or destructor function-try-block handler.
ERR54	Medium	Catch handlers should order their parameter types from most derived to least derived.
ERR55	Low	Honor exception specifications.
ERR58	Low	Handle all exceptions thrown before <code>main()</code> begins executing.
ERR61	Low	Catch exceptions by lvalue reference.
ERR62	Medium	Detect errors when converting a string to a number.
OOP50	Low	Do not invoke virtual functions from constructors or destructors.
OOP52	Low	Do not delete a polymorphic object without a virtual destructor.
OOP53	Medium	Write constructor member initializers in the canonical order.
CON40	Medium	Do not refer to an atomic variable twice in an expression.
CON52	Medium	Prevent data races when accessing bit-fields from multiple threads.
ENV30	Low	Do not modify the object referenced by the return value of certain functions.
ENV32	Medium	All exit handlers must return normally.
ENV33	High	Do not call <code>system()</code> .
FLP30	Low	Do not use floating-point variables as loop counters.
FLP32	Medium	Prevent or detect domain and range errors in math functions.
FLP37	Low	Do not use object representations to compare floating-point values.
MSC30	Medium	Do not use the <code>rand()</code> function for generating pseudorandom numbers.
MSC32	Medium	Properly seed pseudorandom number generators.

MSC33	High	Do not pass invalid data to the <code>asctime()</code> function.
MSC37	High	Ensure that control never reaches the end of a non-void function.
MSC50	Medium	Do not use <code>std::rand()</code> for generating pseudorandom numbers.
MSC51	Medium	Ensure your random number generator is properly seeded.
MSC52	Medium	Value-returning functions must return a value from all exit paths.
MSC53	Medium	Do not return from a function declared <code>[[noreturn]]</code> .
PRE30	Low	Do not create a universal character name through concatenation.
PRE31	Low	Avoid side effects in arguments to unsafe macros.
PRE32	Low	Do not use preprocessor directives in invocations of function-like macros.
SIG31	High	Do not access shared objects in signal handlers.
SIG34	Low	Do not call <code>signal()</code> from within interruptible signal handlers.
SIG35	Low	Do not return from a computational exception signal handler.

Axivion Bauhaus Suite incorporates portions of the “SEI CERT C Coding Standard Wiki” (<https://www.securecoding.cert.org/confluence/display/c/SEI+CERT+C+Coding+Standard>), Copyright © 1995-2016 Carnegie Mellon University, the “SEI CERT C++ Coding Standard Wiki” (<https://www.securecoding.cert.org/confluence/pages/viewpage.action?pageId=637>), Copyright © 1995-2016 Carnegie Mellon University, and the “SEI CERT C Coding Standard: Rules for Developing Safe, Reliable, and Secure Systems” (2016 Edition), Copyright © 2016 Carnegie Mellon University, with special permission from its Software Engineering Institute; however, this Derivative Work has not been reviewed nor is it endorsed by Carnegie Mellon University or its Software Engineering Institute. CERT is a registered trademarks of Carnegie Mellon University.