BB. Definitions

abnormal termination [Open Group 08]
Abnormal termination occurs when requested by the `abort()` function or when some signals are received.

application binary interface
An application binary interface is an interface between two independently compiled modules of a program. An Application Binary Interface document specifies a set of conventions such as the order and location of function arguments that compilers must adhere to in order to achieve interoperability between such modules.

basic exception safety [Stroustrup 01, Sutter 00]
The basic exception safety guarantee is a property of an operation such that, if the operation terminates by raising an exception, it preserves program state invariants and prevents resource leaks. (See also exception safety and strong exception safety.)

Clang
Clang is an open source C and C++ compiler. More information can be found at http://clang.llvm.org/.

condition predicate
A condition predicate is an expression constructed from the variables of a function that must be true for a thread to be allowed to continue execution.

conforming [ISO/IEC 14882-2014]
Conforming programs may depend on nonportable features of a conforming implementation.

critical sections
Critical sections are code that accesses shared data and that would otherwise be protected from data races.

cv-qualify
To cv-qualify a type is to add `const` or `volatile` (or both) to the type.

data race [ISO/IEC 14882 2014]
The execution of a program contains a data race if it contains two potentially concurrent conflicting actions, at least one of which is not atomic, and neither happens before the other. [An exception is] that two accesses to the same object of type `volatile sig_atomic_t` do not result in a data race if both occur in the same thread, even if one or more occurs in a signal handler.

deadlock [ISO/IEC 14882 2014]
A deadlock is when one or more threads are unable to continue execution because each is blocked waiting for one or more of the others to satisfy some condition.

denial-of-service attack
A denial-of-service attack is an attempt to make a computer resource unavailable to its intended users.

diagnostic message [ISO/IEC 14882-2014]
A diagnostic message is a message belonging to an implementation-defined subset of the implementation’s message output. A diagnostic message may indicate a constraint violation or a valid, but questionable, language construct. Messages typically include the file name and line number pointing to the offending code construct. In addition, implementations also often indicate the severity of the problem. Although the C++ Standard does not specify any such requirement, the most severe problems often cause implementations to fail to fully translate a translation unit. Diagnostics output in such cases are termed errors. Other problems may cause implementations simply to issue a warning message and continue translating the rest of the program. (See error message and warning message.)

error message
An error message is a diagnostic message generated when source code is encountered that prevents an implementation from translating a translation unit. (See diagnostic message and warning message.)

exception safety [Stroustrup 01]
An operation on an object is said to be exception safe if that operation leaves the object in a valid state when the operation is terminated by throwing an exception. (See also basic exception safety and strong exception safety.)

exploit [Seacord 2005]
An exploit is a piece of software or technique that takes advantage of a security vulnerability to violate an explicit or implicit security policy.

fatal diagnostic
A fatal diagnostic is a message that causes an implementation not to perform the translation.
**free store** [ISO/IEC 14882-2014]
The free store is storage managed by the C++ allocation and deallocation functions `::operator new(std::size_t), ::operator delete(void*)`, their array forms `::operator new[](std::size_t), ::operator delete[](void*)`, overloads of said functions on `std::nothrow_t`, any user-defined replacements for said functions, as well as any such functions defined as a member of a class. Storage in the free store is distinct from storage managed by the C functions `calloc`, `free`, `malloc`, and `realloc`.

**GCC**
GCC is an open source C and C++ compiler. More information can be found at https://gcc.gnu.org/.

**ill-formed program** [ISO/IEC 14882-2014]
An ill-formed program is a C++ program that is not well-formed; that is, a program not constructed according to the syntax rules, diagnosable semantic rules, and the one-definition rule.

**implementation** [ISO/IEC 9899:2011]
An implementation is a particular set of software, running in a particular translation environment, under particular control options, that performs translation of programs for, and supports execution of functions in, a particular execution environment.

**implementation-defined behavior** [ISO/IEC 14882-2014]
Implementation-defined behavior is behavior, for a well-formed program construct and correct data, that depends on the implementation and that each implementation documents.

**incomplete type** [ISO/IEC 14882-2014]
A class that has been declared but not defined, an enumeration type in certain contexts, an array of unknown size or of incomplete element type, and the void type are incomplete types. These types lack the information required to determine the size of the type.

**indeterminate value** [ISO/IEC 14882-2014]
When storage for an object with automatic or dynamic storage duration is obtained, the object has an indeterminate value, and if no initialization is performed for the object, that object retains an indeterminate value until that value is replaced.

**invalid pointer**
An invalid pointer is a pointer that is not a valid pointer.

**libc++**
libc++ is an open source Standard Template Library (STL) implementation. More information can be found at http://libcxx.llvm.org/.

**libstdc++**
libstdc++ is an open source Standard Template Library (STL) implementation. More information can be found at https://gcc.gnu.org/onlinedocs/libstdc++/

**liveness**
Liveness is when every operation or method invocation executes to completion without interruptions, even if it goes against safety.

**lvalue** [ISO/IEC 14882-2014]
An lvalue (so called, historically, because lvalues could appear on the left-hand side of an assignment expression) designates a function or an object.

**Microsoft Visual Studio**
Microsoft Visual Studio is a commercial C and C++ compiler. More information can be found at https://www.visualstudio.com/.

**Microsoft Visual Studio STL**
The Microsoft Visual Studio STL is a commercial Standard Template Library (STL) implementation. More information can be found at https://msdn.microsoft.com/en-us/library/cc687y.aspx

**mitigation** [Seacord 2005]
A mitigation is a method, technique, process, tool, or runtime library that can prevent or limit exploits against vulnerabilities.

**one-definition rule (ODR)** [ISO/IEC 14882-2014]
A fundamental C++ rule that states that no translation unit shall contain more than one definition of any variable, function, class type, enumeration type or template, and that every program shall contain exactly one definition of every non-inline function or variable. Some definitions may be duplicated in multiple translation units, subject to strict rules.
ODR-use [ISO/IEC 14882-2014]
A function or object is ODR-used if the address of the entity is taken, the function is called, or a reference is bound to the object. When a function or object is ODR-used, its definition must exist within the program or else the program is ill-formed.

RAII (Resource Acquisition Is Initialization)
RAII is a design principle supported by C++. Holding a resource is a class invariant where the allocation of the resource (acquisition) is inseparable from the initialization of the object during its construction. Further, deallocation of the resource is performed during the destruction of the object. Thus, the resource is held when initialization completes and remains held until finalization begins, ensuring there are no resource leaks unless the object owning the resource is also leaked.

reliability [IEEE Std 610.12 1990]
Reliability is the ability of a system or component to perform its required functions under stated conditions for a specified period of time.

restricted sink [ISO/IEC 9899:2011]
A restricted sink is an operand or argument whose domain is a subset of the domain described by its type.

rvalue [ISO/IEC 14882-2014]
An rvalue (so called, historically, because rvalues could appear on the right-hand side of an assignment expression), is an xvalue, a temporary object or subobject thereof, or a value that is not associated with an object.

security flaw [Seacord 2005]
A security flaw is a software defect that poses a potential security risk.

security policy [Internet Society 2000]
A security policy is a set of rules and practices that specify or regulate how a system or organization provides security services to protect sensitive and critical system resources.

strong exception safety [Stroustrup 2001, Sutter 2000]
The strong exception safety guarantee is a property of an operation such that, in addition to satisfying the basic exception safety guarantee, if the operation terminates by raising an exception, it has no observable effects on program state. (See also exception safety and basic exception safety.)

SFINAE
SFINAE (Substitution Failure Is Not An Error) is a language rule applied by the compiler during overload resolution involving templates. In some contexts, when substituting a template type parameter fails, the specialization is discarded from the overload set instead of causing a compile error. This feature is used in template metaprogramming.

tainted source [ISO/IEC TS 17961:2012]
A tainted source is an external source of untrusted data.

tainted value [ISO/IEC TS 17961:2012]
A tainted value is a value derived from a tainted source that has not been sanitized.

trust boundary
A trust boundary is a boundary between a trusted execution context (or trusted data source) in which all sub-execution contexts (or data sources) are trusted by the system and a nontrusted execution context (or nontrusted data sink).

undefined behavior [ISO/IEC 14882-2014]
Undefined behavior is behavior, such as might arise upon use of an erroneous program construct or erroneous data, for which the C++ Standard imposes no requirements. Undefined behavior may also be expected when the C++ Standard omits the description of any explicit definition of behavior or defines the behavior to be ill-formed, with no diagnostic required.

unspecified behavior [ISO/IEC 14882-2014]
Unspecified behavior is behavior, for a well-formed program construct and correct data, that depends on the implementation. The implementation is not required to document which behavior occurs.

unspecified value [ISO/IEC 9899:2011]
An unspecified value is a valid value of the relevant type where the C++ Standard imposes no requirements on which value is chosen in any instance.
**valid pointer**
A valid pointer is a pointer that refers to an element within an array or one past the last element of an array. For the purposes of this definition, a pointer to an object that is not an element of an array behaves the same as a pointer to the first element of an array of length one with the type of the object as its element type. For the purposes of this definition, an object can be considered to be an array of a certain number of bytes; that number is the size of the object, as produced by the `sizeof` operator.

**vtable**
A vtable is a common implementation technique to support dynamic method dispatch, where a class object instance includes a hidden data member that is a pointer to an array of function pointers used to resolve virtual function calls at runtime.

**vulnerability** [Seacord 2005]
A vulnerability is a set of conditions that allows an attacker to violate an explicit or implicit security policy.

**warning message**
A warning message is a diagnostic message generated when source code is encountered that does not prevent an implementation from translating a translation unit. (See diagnostic message and error message.)

**well-formed program** [ISO/IEC 14882-2014]
A well-formed program is a C++ program constructed according to the syntax rules, diagnosable semantic rules, and the one-definition rule. (See also ill-formed program.)

**xvalue** [ISO/IEC 14882-2014]
An xvalue (an “eXpiring” value) also refers to an object, usually near the end of its lifetime (so that its resources may be moved, for example). An xvalue is the result of certain kinds of expressions involving rvalue references.