Rule	Description	Severity	Likelihood	Remediation Cost	Priority
DCL51	Do not declare or define a reserved identifier	Low	Unlikely	Low	P3
DCL52	Never qualify a reference type with const or volatile	Low	Unlikely	Low	P3
DCL60	Obey the one-definition rule	High	Unlikely	High	P3
EXP52	Do not rely on side effects in unevaluated operands	Low	Unlikely	Low	P3
CRT58	Predicate function objects should not be mutable	Low	Unlikely	High	P3
ERR61	Catch exceptions by lvalue reference	Low	Unlikely	Low	P3
DCL53	Do not write syntactically ambiguous declarations	Low	Unlikely	Medium	P2
DCL56	Avoid cycles during initialization of static objects	Low	Unlikely	Medium	P2
EXP51	Do not delete an array through a pointer of the incorrect type	Low	Unlikely	Medium	P2
EXP56	Do not call a function with a mismatched language linkage	Low	Unlikely	Medium	P2
CTR57	Provide a valid ordering predicate	Low	Probable	High	P2
ERR53	Do not reference base classes or class data members in a constructor or destructor function-try-block handler	Low	Unlikely	Medium	P2
ERR57	Do not leak resources when handling exceptions	Low	Probable	High	P2
OOP50	Do not invoke virtual functions from constructors or destructors	Low	Unlikely	Medium	P2
OOP54	Gracefully handle self-copy assignment	Low	Probable	High	P2
OOP56	Honor replacement handler requirements	Low	Probable	High	P2
CON54	Wrap functions that can spuriously wake up in a loop	Low	Unlikely	Medium	P2
CON55	Preserve thread safety and liveness when using condition variables	Low	Unlikely	Medium	P2
MSC53	Do not return from a function declared [[noreturn]]	Medium	Unlikely	Low	P2
DCL55	Avoid information leakage when passing a class object across a trust boundary	Low	Unlikely	High	P1
CON56	Do not speculatively lock a non-recursive mutex that is already owned by the calling thread	Low	Unlikely	High	P1