EXP14-C. Beware of integer promotion when performing bitwise operations on integer types smaller than int

Deprecated

This guideline has been deprecated by

- INT02-C. Understand integer conversion rules

Integer types smaller than `int` are promoted when an operation is performed on them. If all values of the original type can be represented as an `int`, the value of the smaller type is converted to an `int`; otherwise, it is converted to an `unsigned int` (see INT02-C. Understand integer conversion rules). If the conversion is to a wider type, the original value is zero-extended for unsigned values or sign-extended for signed types. Consequently, bitwise operations on integer types smaller than `int` may have unexpected results.

Noncompliant Code Example

This noncompliant code example demonstrates how performing bitwise operations on integer types smaller than `int` may have unexpected results.

```
uint8_t port = 0x5a;
uint8_t result_8 = ( ~port ) >> 4;
```

In this example, a bitwise complement of `port` is first computed and then shifted 4 bits to the right. If both of these operations are performed on an 8-bit unsigned integer, then `result_8` will have the value `0x0a`. However, `port` is first promoted to a `signed int`, with the following results (on a typical architecture where type `int` is 32 bits wide):

Expression	Type	Value	Notes
`port`	`uint8_t`	`0x5a`	
`~port`	`int`	`0xffffffa5`	
`~port >> 4`	`int`	`0x0ffffffa`	Whether or not value is negative is implementation-defined.
`result_8`	`uint8_t`	`0xfa`	

Compliant Solution

In this compliant solution, the bitwise complement of `port` is converted back to 8 bits. Consequently, `result_8` is assigned the expected value of `0x0aU`.

```
uint8_t port = 0x5a;
uint8_t result_8 = (uint8_t) (~port) >> 4;
```

Risk Assessment

Bitwise operations on shorts and chars can produce incorrect data.

Recommendation	Severity	Likelihood	Remediation Cost	Priority	Level
EXP14-C	low	likely	high	P3	L3

Automated Detection

Tool	Version	Checker	Description
Axivion Bauhaus Suite	6.9.0	**CertC-EXP14**	Fully implemented
CodeSonar	5.0p0	**LANG.CAST.RIP**	Risky integer promotion
Compass/ROSE			
ECLAIR	1.2	**CC2.EXP14**	Fully implemented
Parasoft C/C++test	10.4.2	**CERT_C-EXP14-a**	Avoid mixing arithmetic of different precisions in the same expression

Related Vulnerabilities

Search for vulnerabilities resulting from the violation of this rule on the CERT website.

Related Guidelines

SEI CERT C++ Coding Standard	VOID EXP15-CPP. Beware of integer promotion when performing bitwise operations on chars or shorts
MISRA-C	Rule 10.5