ISO/IEC JTC 1/SC 22/WG 23 N 0385

Proposed rewrite of Ruby.52

Date	8 January 2012
Contributed by	Jim Moore
Original file name	ProposedRewriteOfRuby-52.docx
Notes	

In document N0378, the current description of Ruby.52 is:

Ruby.52 Provision of Inherently Unsafe Operations [SKL]

This vulnerability does not apply to Ruby. It provides a means for "type-casting" which is safe by honoring classes with the same interface as the same type. If differences are exposed an exception will be raised by the processor.

At the most recent meeting of WG23, I (Jim Moore) was tasked to "come up with a more suitable explanation using references to the main document."
The main document includes this description of SKL:

6.52 Provision of Inherently Unsafe Operations [SKL]

6.52.1 Description of application vulnerability

Languages define semantic rules to be obeyed by conforming programs. Compilers enforce these rules and diagnose violating programs.

A canonical example are the rules of type checking, intended among other reasons to prevent semantically incorrect assignments, such as characters to pointers, meter to feet, euro to dollar, real numbers to booleans, or complex numbers to two-dimensional coordinates.

Occasionally there arises a need to step outside the rules of the type model to achieve needed functionally. One such situation is the casting of memory as part of the implementation of a heap allocator to the type of object for which the memory is allocated. A type-safe assignment is impossible for this functionality. Thus, a capability for unchecked "type casting" between arbitrary types to interpret the bits in a different fashion is a necessary but inherently unsafe operation, without which the type-safe allocator cannot be programmed.

Another example is the provision of operations known to be inherently unsafe, such as the deallocation of heap memory without prevention of dangling references.

A third example is any interfacing with another language, since the checks ensuring type-safeness rarely extend across language boundaries.

These inherently unsafe operations constitute a vulnerability, since they can (and will) be used by programmers in situations where their use is neither necessary nor appropriate.

The vulnerability is eminently exploitable to violate program security.

6.52.2 Cross reference

[None]

6.52.3 Mechanism of Failure

The use of inherently unsafe operations or the suppression of checking circumvents the features that are normally applied to ensure safe execution. Control flow, data values, and memory accesses can be corrupted as a consequence. See the respective vulnerabilities resulting from such corruption.

6.52.4 Applicable language characteristics

This vulnerability description is intended to be applicable to languages with the following characteristics:
• Languages that allow compile-time checks for the prevention of vulnerabilities to be suppressed by compiler or interpreter options or by language constructs, or
• Languages that provide inherently unsafe operations

6.52.5 Avoiding the vulnerability

Software developers can avoid the vulnerability or mitigate its ill effects in the following ways:
• Restrict the suppression of compile-time checks to where the suppression is functionally essential.
• Use inherently unsafe operations only when they are functionally essential.
• Clearly identify program code that suppresses checks or uses unsafe operations. This permits the focusing of review effort to examine whether the function could be performed in a safer manner.

Proposed rewrite:

Ruby.52 Provision of Inherently Unsafe Operations [SKL]

This vulnerability does not apply to Ruby. It provides a means for "type-casting" which is safe by honoring classes with the same interface as the same type. If differences are exposed an exception will be raised by the processor. However, unlike statically typed languages, type safety in a Ruby program cannot be checked at compile-time. Therefore, maintenance of type safety in a Ruby program critically depends upon the correct coding of exception handlers to catch and treat type exceptions. Hence, Ruby programmers must pay particular attention to the class of vulnerabilities described in 6.38 and Ruby.38 [OYB].