Terms of Reference: ISO/IEC Project 22.24772, "Guidance to Avoiding Vulnerabilities in Programming Languages through Language Selection and Use"

Jim Moore
Convener, ISO/IEC JTC 1/SC 22/OWG Vulnerability
James.W.Moore@ieee.org

Project Terms of Reference

- The next several slides describe the “terms of reference” – i.e. scope, purpose, products, etc – of this standards project.
- The terms are derived from the New Work Item Proposal that was approved by SC22.
- I have restated the terms in the form of an FAQ that can be added to our web site.
- Each of the following slides has a common form ...

FAQ:

This box contains a (possible) Frequently Asked Question.

Quote from NP:

This box quotes relevant source material from the approved New Work Item Proposal.

FAQ Response:

This box proposes an answer to the FAQ.

FAQ:

What is the intent of this standards effort?

Quote from NP:

Title
Guidance to Avoiding Vulnerabilities in Programming Languages through Language Selection and Use

Purpose and justification

… so that application developers will be better informed regarding the vulnerabilities inherent to candidate languages and the costs of avoiding such vulnerabilities. An additional benefit is that developers will be better prepared to select tooling to assist in the evaluation and avoidance of vulnerabilities.

FAQ Response:

The intent of the project is to write a report containing guidance to users of programming languages on how to avoid the *vulnerabilities* that exist in the programming language selected for a particular project. Implicitly, the guidance may also be helpful in selecting a language for a particular project. Finally, the report may be helpful in choosing tooling to assist in evaluating and avoiding vulnerabilities.

FAQ:

What is a “vulnerability”?

Quote from NP:

FAQ Response:

[Tentative – subject to discussion.]

“A flaw in a product that makes it infeasible, even when using the product properly, to prevent an attacker from usurping privileges on the user's system, regulating its operation, compromising data on it, or assuming ungranted trust.”

-- From Microsoft, “Definition of a Security Vulnerability”:

http://www.microsoft.com/technet/archive/community/columns/security/essays/vulnrbl.mspx?mfr=true

FAQ:

What is “guidance”?

Quote from NP:

FAQ Response:

Some standards documents contain "normative" provisions – requirements that a user must obey in order to be allowed to make a claim of conformance to the standard. This document will not contain such provisions; it will contain only information and suggestions for users.

FAQ:

Is the guidance intended to be “one size fits all”?

Quote from NP:

Scope The guidance could be applicable to any software development project applying the programming languages considered in the TR. The advisability of applying the guidance would vary depending upon the criticality of properties such as safety, security or privacy. ...

FAQ Response:

No. When one considers a number of vulnerabilities, the severity of their consequences may vary and the resource necessary to avoid them or mitigate their consequences may also vary. Users need to make an informed selection of vulnerabilities to be treated based on the need of the software product to preserve *critical properties* and the cost of preserving those properties.

FAQ:

What is a “critical property”?

Quote from NP:

Purpose and justification ... These problems can have serious consequences for systems that are intended to implement integrity properties such as safety, security or privacy. Although the consequences may be less severe, there is also the cost of dealing with electronic vandalism enabled by vulnerabilities in programs that are not themselves intended to have high integrity properties.

FAQ Response:

Of course, we hope that all software works correctly. However, consequences of software failure can be particularly severe when the failure compromises safety, security, or privacy. These are examples of critical properties. A critical property is a property of software and its containing system that must be preserved in order to avoid failures that present unacceptably severe consequences.

FAQ:

Is there more critical software now than in the good old days?

Quote from NP:

FAQ Response:

Well, there's the obvious fact that software is now being used in more devices, including devices with critical properties. In addition to this, the increasing connectivity provided by the internet increases the vulnerability of all software. An intruder can attack any piece of software that is executing on a machine, and then use that software as a springboard to attack other software on the machine or on the network. So it becomes important to decrease the vulnerability of ***all*** software – even software with low criticality.

FAQ:

So, is all software critical?

Quote from NP:

FAQ Response:

No, there will always be software where additional development effort is justified to increase safety, security, or privacy. However, it is now appropriate for all software (except software in isolated, dedicated systems) to exhibit greater resistance to attack and exploitation.

FAQ:

What sorts of vulnerabilities exist in programming languages?

Quote from NP:

Purpose and justification - Any programming language contains constructs that are vague or difficult to use. Many language definitions include "implementation dependencies" that can affect their semantics in different execution environments. There is a set of "common mode" failures that occur across a variety of languages. Finally, there are weaknesses in language constructs that can be exploited by attackers, for example, the now-famous "buffer overrun" attacks. As a result, software programs sometimes execute differently than was intended by their developers.

FAQ Response:

Any programming language contains constructs that are vague or difficult to use. Many language definitions include "implementation dependencies" that can affect their semantics in different execution environments. There is a set of "common mode" failures that occur across a variety of languages, for example, problems with pointers. Finally, there are weaknesses in constructs provided by particular languages, for example, "buffer overrun" in C. There was a time when many of these weaknesses might have been regarded as benign. In a connected world, though, the weaknesses can be exploited by attackers. "Buffer overrun" is simply the most famous example.

FAQ:

Is there a criterion for selecting language features for guidance?

Quote from NP:

Purpose and justification -

... Successful treatment of these problems would result in the production of software codes that exhibit more predictable behaviour in execution. ... One criteria for selecting guidance for the report would be whether the guidance improves the predictability of execution.

FAQ Response:

The general idea is that a particular program code should execute the behaviour that the programmer intended, even when stimulated in unanticipated ways by external parties or events. We call this property "predictable execution". We will consider providing guidance on a feature of a programming language if that guidance can improve the predictability of execution of program codes containing the feature.

FAQ:

Is it possible to completely protect code from exploitation by attackers?

Quote from NP:

Purpose and justification

... Although an ideal result is currently impractical, "predictable execution" is an ideal toward which we can strive.

FAQ Response:

The goal is to provide guidance helping coders improve the predictability of the execution of their code, even in the face of attempted exploitation. Complete success is infeasible, so the goal of fully predictable execution represents an ideal. Nevertheless, practical guidance can help to reduce the frequency and the consequences of successful attack.

FAQ:

How will the Technical Report be organized?

Quote from NP:

Purpose and justification

... The purpose of this project is to prepare comparative guidance spanning a large number of programming languages ...

FAQ Response:

The plan is to organize the report by type of vulnerability. Then the Technical Report would describe how each type of vulnerability manifests itself in the various programming languages. For each language the report would provide techniques for avoiding the vulnerability or guidance for mitigating its ill effects.

FAQ:

What sort of guidance will be provided?

Quote from NP:

Purpose and justification

... In developing the guidance, the project will prefer linguistic means of avoiding vulnerabilities but, when necessary may describe extra-linguistic means (e.g. static analysis or targeted testing). In developing the guidance, the project will prefer the avoidance of identified risks but, when necessary, may describe means to mitigate the risk of vulnerabilities that cannot be economically avoided. Finally, in cases where identified problems can be neither avoided nor mitigated, the report may assist users in understanding the nature of risk that must be accepted.

FAQ Response:

In the simplest of cases, the report will suggest alternative coding patterns that are equally effective but which avoid the vulnerability or which otherwise improve the predictability of execution. When that is not possible, the report may suggest the use of static analysis techniques and may provide guidance for coding in a manner that will improve the effectiveness of the analysis. When static analysis is not feasible, the report may suggest the use of other testing or verification techniques. Whenever possible, the report will assist users in understanding the cost and benefits of avoiding risks and the nature of any residual risks.

FAQ:

Is this guidance only for coders?

Quote from NP:

Purpose and justification

... The admission that some problems must be treated via analysis or testing introduces a secondary consideration in recommending linguistic means for avoiding vulnerabilities; in some situations, one construct might be preferred over another on the grounds that it is easier to test or easier to analyze. This relationship between construction and subsequent verification activities makes it clear that the report will be useful both for those emphasizing "correctness by construction" and those who desire to improve the predictability of execution through testing and analysis.

FAQ Response:

No. In some cases, it may be difficult to avoid the vulnerability through coding techniques. One might have to rely on static analysis or testing in order to evaluate the vulnerability. In some cases, though, naively coded constructs cannot be effectively analyzed. In such cases, it may be preferable to design the program in a particular way – not because it avoids the vulnerability – but because it allows the analysis or testing to be performed more effectively. A balanced approach to dealing with the vulnerabilities will involve design, construction, testing and verification.

FAQ:

What are the planned products of the project?

Quote from NP:

Programme of work

If the proposed new work item is approved , which of the following document(s) is (are) expected to be developed? ... a technical report, type 3

Scope ... In addition to producing a Technical Report, it is possible that the working group might create recommendations for working groups that maintain the standards or specifications for the programming languages considered in the TR.

FAQ Response:

Two products are envisioned. One is a Type 3 Technical Report intended for users of programming languages. The Technical Report will contain guidance explaining different kinds of vulnerabilities and how they can be avoided in different programming languages. The second product might be feedback to the standards committees responsible for programming language standards, suggesting changes that could be made to their language specifications.

FAQ:

What is a Type 3 Technical Report?

Quote from NP:

FAQ Response:

The most important point is that it is *not* a standard. ISO/IEC JTC 1 develops two types of documents: standards and technical reports—three types of them. A Type 3 Technical Report is a document that is *inherently unsuitable* to be a standard. The planned document precisely fits that category because it will not include normative provisions.

FAQ:

Will the report be guided by empirical data?

Quote from NP:

Purpose and justification
... Although a strict reliance on empirical evidence of effectiveness and quantified analysis of cost/benefit is not feasible, the project will be guided by both of those notions in its selection of guidance to be included in the report. Because of the dearth of quantifiable evidence, a cautious approach to incorporating guidance may be appropriate.

FAQ Response:

Ideally, yes. However, in many areas empirical data is simply not available and the collective judgment of experts may be used.

FAQ:

Where will the list of vulnerabilities come from?

Quote from NP:

FAQ Response:

Currently, there are non-ISO efforts underway called "Common Vulnerability Enumeration" and "Common Weakness Evaluation". These projects involve a partnership among tool makers and users to provide common names to vulnerabilities and weaknesses, permitting parties to communicate about their nature. A representative of these projects, Bob Martin, will attend the June meeting so that we can determine whether these projects might provide the vulnerability data that we need. If all else fails, we might have to develop an ad hoc list based on the judgment of the working group.

FAQ:

What languages will be considered?

Quote from NP:

Relevant documents to be considered

- The programming language standards of ISO/IEC JTC 1/SC 22.
- For market reasons, the specifications of popular languages that are not the subject of ISO standards. ...
- Existing documents providing usage guidance for individual languages, e.g. ISO/IEC TR 15942, MISRA C, NUREG/CR-6463. ...
- Existing work on safe programming approaches, e.g. the SPARK language, ISO/IEC draft TR 24731 (Specification for Secure C Library Functions)

FAQ Response:

Any of the programming languages maintained by JTC 1/SC 22 are candidates. Also, popular languages maintained by groups outside ISO are also candidates, e.g. C# and Java. The working group will have to make a cost/benefit judgment regarding which languages to include. This judgment will depend, in part, upon the ability to obtain participation by experts in those languages. Currently, the group is not considering scripting languages, like Python, or design languages, like UML.

FAQ:

What other work will be considered?

Quote from NP:

Relevant documents to be considered

- ... The software engineering standards of ISO/IEC JTC 1/SC 7, as a source of extra-linguistic mitigation methods. ...
- Standards dealing with functional safety, notably, IEC 61508. ...

FAQ Response:

In some cases, programming language vulnerabilities cannot be feasibly avoided. In such cases, one might apply other software engineering, risk management, or safety engineering techniques to deal with the problem. The software engineering standards of ISO/IEC JTC 1/SC 7 may be relevant in this area, as well as the functional safety standard, IEC 61508.

FAQ:

Who will participate in the working group?

Quote from NP:

Cooperation and liaison

The proposal recognizes that a normally constituted working group will not suffice to perform the necessary work. In addition, to the usual National Body participants, it is proposed to use experts appointed by each existing working group in JTC 1/SC 22, liaison experts appointed by other organizations maintaining programming language specifications, and liaison experts appointed by other standards committees maintaining related documents.

FAQ Response:

As is normally the case, the Technical Report will have to be approved by a consensus of national bodies. Therefore, participants in the working group will be representatives of those NBs. However, it is clear that this group will need additional sources of expertise. Therefore, the working groups of SC22 have also been asked to name experts to participate in the group. We are also seeking to establish liaisons to provide experts for non-ISO languages.

FAQ:

What is the schedule for doing the work?

Quote from NP:

Preparatory work offered with target date(s)

... It is proposed to perform the work on the "normal" (36 month) schedule. However, it is recognized that the work may be performed in increments that subdivide the schedule or with refinements that would produce subsequent amendments or revisions.

FAQ Response:

The tentative schedule for the work is:

First meeting: June 2006

Document outline: Jan 2007

Submission to SC 22: Jan 2008

Submission to JTC 1: Oct 2008

Publication: Jan 2009

FAQ:

Where can I obtain more information?

Quote from NP:

Preparatory work offered with target date(s)

A web site provides a summary of progress to date:
http://www.aitcnet.org/isai/

…

FAQ Response:

A web site provides a summary of progress to date:
http://www.aitcnet.org/isai/

Some Examples from Existing Documents

An example of "comparative guidance"

- From the NP: *... prepare comparative guidance spanning a large number of programming languages*

Example from NUREG/CR-6463, Rev. 1, ***Review Guidelines for Software Languages for Use in Nuclear Power Plant Safety Systems: Final Report***, 1997, US Nuclear Regulatory Commission

If dynamic memory allocation is unavoidable, the source code should include provisions to ensure that:

- All dynamically allocated memory during a specific execution cycle is released at the end of that cycle, and
- The possibility of interruption of execution between the point where memory is dynamically allocated and when it is released is minimized (if not totally eliminated); there should also be provisions in the application code that will detect any situation where dynamically allocated memory has not been released and release such memory.

To see the following languages select: Ada; C and C++ ; Pascal; PL/M; Ada 95.

Example from NUREG/CR-6463, Rev. 1, ***Review Guidelines for Software Languages for Use in Nuclear Power Plant Safety Systems: Final Report***, 1997, US Nuclear Regulatory Commission

The following discussion applies to C++ only.

In C++, the functions to dynamically allocate and free memory are new and delete. The following guideline applies.

- Ensure that all classes include a destructor. To avoid memory leaks, all classes must include a destructor that releases any memory allocated by the class. Constructors must themselves be defined in a way to avoid possible memory leaks in case of failures. Ensure that for all derived classes there are virtual destructors.

An example of the relationship of coding and analysis

- From the NP: *... the project will prefer linguistic means of avoiding vulnerabilities but, when necessary may describe extra-linguistic means (e.g. static analysis or targeted testing) ...*

Example from ISO/IEC TR 15942:2000, ***Information technology — Programming languages — Guide for the use of the Ada programming language in high integrity systems***

Initialization of Variables

All variables should be given a meaningful value before use. Failure to do so may raise a predefined exception or cause a bounded error at run-time. Initial values may be given by:

1. Associating an explicit initialization expression with the variable at the point of its declaration.
2. Making an assignment to the variable that will be executed prior to references to it.

For state variables in packages, assignments may also be made in the package elaboration part. A consistent approach to the initialization of package state variables should be adopted.

In all cases, Data Flow analysis should be used to confirm that every object has been assigned a value before it is used. The effectiveness of the analysis is undermined if variables are initialized unnecessarily (sometimes called 'junk initialization'). ...