Programming Language Vulnerabilities within the ISO/IEC Standardization Community

Stephen Michell

International Convenor JTC 1/SC 22 WG 23
Programming Language Vulnerabilities

Canadian HoD to ISO/IEC/JTC 1SC 22
Programming Language Subcommittee

stephen.michell@maurya.on.ca

Meet JTC 1

- ISO is the International Standards Organization
- IEC is the International Electrotechnical Commission
    - Both have international treaties to develop International Standards
    - Both work through internationally manned Technical Committees to develop standards
        - e.g.
            - ISO 9001 Quality
            - IEC 61508 Safety
    - Why? - International standards can be readily adopted by countries and put into national regulations.
    - Work is done by consensus
        - Wide agreement, no strong sustained opposition

JTC 1

- ISO and IEC jointly formed the Joint Technical Committee 1(circa 1988)
    - Everything IT
        - Printers, media, network protocols, databases. software engineering, big data
        - and oh yes
            - programming languages
    - Has own procedures and Subcommittees to do the work

Meet Subcommittee(SC) 22

- Programming languages and their environments

APL COBOL Fortran
Basic Mumps POSIX
Pascal Ada Internationalization
C Lisp Prolog
Modula 2 Formal Methods
C++ Vulnerabilities

Meet SC 22 (cont)

- Member Countries (20 P members)

Austria	Canada	China
France	Denmark	Germany
Japan	Korea	Spain
Switzerland	USA	UK

and others that are not usually in plenary

- Also O Members

Belgium	New Zealand	Singapore
India	Italy	Argentina

How Standardized?

- National Body (NB) participation and voting
- Project steps
  - New Work Item Proposal (NB approval)
  - Working Draft (technical expert consensus)
  - Committee Draft (national body consensus)
  - Draft International Standard (JTC 1 vote)
  - Standard
- Countries provide technical experts that do the work
- Documents iterate through the projects steps with international votes
  - Last one (FDIS) -> Standard!

How Standardized?

- Also produce other international products
  - Technical Corrigendum to standard
  - Amendment to standard
  - Technical Specification (pre-standard)
  - Technical Report

What about innovation?

- Working with some of the best in the world
- Adding new capabilities and ideas as they mature enough to standardize
  - Interfaces, Containers (Ada)
  - Assertions, Ravenscar profile (Ada)
  - Bounded Libraries(C)
  - Concurrency features, Static Assertions (C)
  - Parallelism (fine-grained) (Ada, C, C++, Fortran)
  - Concepts, Lambdas (C++)
  - Async methods (C#, C++)
  - Interfacing to C (Fortran)
  - OO (Fortran, COBOL)

Programming Language Vulnerabilities (WG 23)

- Develop a Technical Report on language independent vulnerabilities with language-dependent annexes to map each language to the common ones.
  - Published as TR 24772:2010
  - Revised 2012 with annexes for C, Ada, Ruby, Python, Spark and PHP.
  - Revising TR 24772 to add more vulnerabilities (OO, Time) and more languages (Fortran, C++)
- Published FDIS 17960 Code Signing for Source Code

Outreach

- Work with other groups
  - ISO/IEC/JTC 1/SC 27 Security (liaison)
  - Programming language WG's (WG 9 Ada, WG 14 C, WG 5 Fortran, etc)
  - IEC SC 65 for Safety (liaison being initiated)

Vulnerabilities

- Various groups look at programming language vulnerabilities
  - MITRE/Homeland Security
    - Common Vulnerabilities and Exposures (CVE)
      - Enumerates every vulnerability instance reported by type, OS, application (thousands)
    - Common Weakness Exposures (CWE)
      - Groups reported vulnerabilities by type (about 900)
      - SANS/CWE Top 10
    - Open Wasp Application Project
      - OWASP Top 25

Vulnerabilities (WG 23)

- Different look at vulnerabilities
  - More than Security – Safety also
  - Consider much more than attacks
    - Programming mistakes
      - From classic to obscure
      - Consider real time issues
    - Weaknesses that can be attacked
  - Aggregated more than CWE
    - Document about 90 vulnerabilities that match 900 CWE weaknesses
  - Consider how vulnerabilities appear in specific programming languages
    - Separate annex for each programming language

What WG 23 has not done

- Coding Standards
  - Many levels of integrity (safety and security) will use this document
  - Many programming domains will use documents, from general usage to real time community
  - Concerns of each community is different and the ways that they address vulnerabilities will differ
  - No hope that a single coding standard will meet the needs of any (let alone all) community
  - Writing to the people that create coding standards
  - WG 23, however, is consolidating common guidance that many will use as coding guidelines

Vulnerabilities (WG 23)

- Intend that document will be used to develop coding standards
- Provide explicit guidance to programmer to avoid vulnerability
  - Use static analysis tools
  - Adopt specific coding conventions
  - Always check for error return
- Recommend to language designers on steps to eliminate vulnerability from language
  - Provide move/copy/etc operations that obey buffer size and boundaries

Vulnerabilities (WG 23)

- Vulnerabilities covered
    - Type system
    - Bit representation
    - Floating point arithmetic
    - Enumeration issues
    - Numeric conversion issues
    - String termination Issues
    - Buffer boundary violations
    - Unchecked array indexing
    - Unchecked array copying
    - Pointer type changes
    - Pointer arithmetic
    - Null pointer dereference

Vulnerabilities (WG 23)

- Vulnerabilities covered (more)
  - Identifier name reuse
  - Unused variable
  - Operator precedence / order of evaluation
  - Switch statements and static analysis
  - Ignored status return and unhandled exceptions
  - OO Issues (overloading, inheritance, etc)
  - Concurrency Issues (activation, directed termination, premature termination, concurrent data access)
  - Time Issues (time jumps, jitter, representation)

Vulnerabilities (WG 23)
Application Vulnerabilities

- Design errors that cannot be traced to language weaknesses
  - Adherence to least privilege (not)
  - Loading/executing untrusted code
  - Unrestricted file upload
  - Resource exhaustion
  - Cross site scripting
  - Hard coded password
  - Insufficiently protected credentials

Vulnerabilities (WG 23)

- Look at one vulnerability
  - 6.5 Enumerator Issues [CCB]
  - 6.5.1 Description of Vulnerability
    - What is enumeration
    - Issue of non-default representation, duplicate values,
    - Issue of arrays indexed by enumerations
      - Holes
    - Issue of static coverage
  - 6.5.2 References
    - Reference
      - CWE counterpart,
      - MISRA C and C++ rules,
      - CERT C guidelines,
      - JSF AV rules,
      - Ada Quality Style and Guide

Vulnerabilities (WG 23)

- 6.5.3 Mechanism of Failure
    - Interplay between order of enumerators in list, how (and where) new members added, and changes in representation.
    - Expressions that depend on any of these are fragile
        - Incorrect assumptions can lead to unbounded behaviours
- 6.5.4 Applicable Language Characteristics
    - Languages that permit incomplete mappings (to theoretical enumeration)
    - Languages that provide only mapping of integer to enumerator
    - Languages that have no enumerator capability

Vulnerabilities (WG 23)

- 6.5.5 Avoiding Vulnerability & Mitigating Effects
  - Use static analysis tools to detect problematic use
  - Ensure coverage of all enumeration values
  - Use enumeration types selected from limited set of values
- 6.5.6 Implications for Standardization
  - Provide a mechanism to prevent arithmetic operations on enumeration types
  - Provide mechanisms to enforce static matching between enumerator definitions and initialization expressions

Vulnerabilities (WG 23)

- Ada’s response to Enumerator Issues
  - Complete coverage mandatory
  - Order must be preserved, but holes in representation permitted
  - Arrays indexed by enumeration type may have holes (implementation dependent)
  - When “others” option used in enumeration choice, unintended consequences can occur
  - Guidance
    - Do not use “others” choice for case statements & aggregates
    - Mistrust subranges as choices after enumeration values added in middle

Vulnerabilities (WG 23)

- C’s response on Enumerator Issues
  - Follow guidance of main part
  - Use enumerators starting at 0 and incrementing by 1
  - Avoid loops that step over enumerator with non-default representation
  - Select from limited set of choices, and use static analysis tools

Vulnerabilities (WG 23)

- Python’s response on Enumerator Issues
  - Python only has named integers and sets of strings
  - Variable can be rebound at any time, so no consistent use as an enumerator

Vulnerabilities (WG 23)

- First version of TR 24772 published in 2010
  - No language specific annexes ready
- Second edition published in 2012
  - Language annexes for Ada, C, Python, Ruby, Spark, PHP
  - New vulnerabilities for concurrency but no language-specific response

Vulnerabilities (WG 23)

- Ongoing work
  - Separate 1 document into main part (24772-1) and language-specific parts (Ada -2, C -3, etc)
    - Simplifies maintenance
  - Add more language-specific annexes
    - Fortran   Java   C++   COBOL
  - Add writeups for concurrency vulnerabilities in language-specific annexes
  - Improve a number of vulnerability writeups

Vulnerabilities (WG 23)

- Ongoing Work (cont)
  - Add vulnerabilities
    - Floating point
      - Have one, but very general
    - Object Orientation
      - Examination of C++, etc, show missing areas
    - Time
  - Consider application-level vulnerabilities
    - Have we addressed issues such as “heartbleed”?
  - Think about coding standards and design standards for application-level vulnerabilities
  - Consider creation of top-10/12 avoidance techniques

Contact

- Programming Languages is an exciting field, especially in a world of “too many cores”.
- If you are interested in programming languages or standardization in general,
  - Your National body representative
  - Or me,
    stephen.michell@maurya.on.ca