Programming Language Vulnerabilities within the ISO/IEC Standardization Community

Stephen Michell

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

stephen.michell@maurya.on.ca

WG 23

The Programming Languages Working Group is officially the

ISO IEC

Joint Technical Committee 1

Subcommittee 22 (Programming Languages)

Working Group 23 on Programming Language Vulnerabilities

Programming Language Vulnerabilities (WG 23)

- Develop a Technical Report on programming language independent vulnerabilities with language-dependent annexes to map each language to the common ones.
- Published as TR 24772:2010
- Revised 2013 with annexes for C, Ada, Ruby, Python, Spark and PHP.
- Published IS 17960 Code Signing for Source Code

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

How is WG 23 Different?

- Different look at vulnerabilities
  - Consider much more than attacks
    - Programming mistakes
      - From classic to obscure
      - Consider real time issues
      - Safety-related and security-related
    - Weaknesses that can be attacked
  - Aggregated more than CWE
    - Document about 90 vulnerabilities that cover 900 CWE weaknesses
  - Consider how vulnerabilities manifest in programming languages
    - Separate “part” 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

Vulnerabilities (WG 23)

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

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

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 (activation, termination, data access)

Vulnerabilities (WG 23)
Application Vulnerabilities

- Design errors (can't be traced to language weaknesses)
  - Lack of adherence to least privilege
  - Loading/executing untrusted code
  - Unrestricted file upload
  - Resource exhaustion
  - Cross site scripting
  - Hard coded password
  - Insufficiently protected credentials
  - Time and clock-based vulnerabilities (new)

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

Changes Since 2012

- Split Language-dependent parts from Language Independent part
  - TR 24772-1 Guidance – language independent
  - TR 24772-2 Guidance – Ada
  - TR 24772-3 Guidance – C
  - TR 24772-4 Guidance – Python
  - TR 24772-5 Guidance – PHP
  - TR 24772-6 Guidance – Spark
  - TR 24772-7 Guidance – Ruby
  - TR 24772-8 Guidance – Fortran

Changes Since 2012

- Add 7 new vulnerabilities
  - Object Orientation (clause 6, language-based)
    - Deep vs shallow copying
    - Violations of Liskov principle (contract model)
    - Redispatching
    - Polymorphic Variables
  - Time and Clock (clause 7 application)
    - Time consumption
    - Clock Issues
    - Time drift and jitter

Changes Since 2012

- Guidance for each vulnerability is more “directive”
- Aggregate guidance into a “top 10” list for part
  - Validate Input
  - Check return values
  - Do static analysis
- Why?
  - Gives
    - Smaller footprint for solution space
    - Easier entry into the meat of the document, the explanations.

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

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
  - Arithmetic operations on enumerations
- 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
    - Compiler (e.g.Ada) can be useful
  - 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 rules on 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 "when others =>" option used in enumeration choice, unintended consequences
  - Guidance
    - Do not use "others" choice for case statements & aggregates
    - Mistrust subranges as choices (enumeration values added in middle)

Vulnerabilities (WG 23)

- C's guidance 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 take 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

Contact

- Programming Languages is an exciting field, especially in a world of “too many cores”.
- If you are interested in participating in programming language vulnerabilites, especially if you know Ruby, Java, C#, EcmaScript,
  - Contact me,
    stephen.michell@maurya.on.ca