A New Standards Project on “Avoiding Programming Language Vulnerabilities”

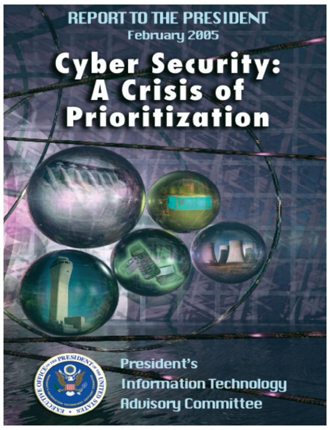
Jim Moore
Liaison Representative from IEEE Computer Society to ISO/IEC JTC 1/SC 7
Liaison Representative between ISO/IEC JTC 1/SC 7 and SC 22
Convener, ISO/IEC JTC 1/SC 22/OWG Vulnerability
James.W.Moore@ieee.org

Cyber Security is a Growing Problem

President’s Information Technology Advisory Committee (PITAC) Subcommittee on Cyber Security

Areas in Need of Increased Support

- Computer Authentication Methodologies
- Securing Fundamental Protocols
- ***Secure Software Engineering and Software Assurance***
- Holistic System Security
- Monitoring and Detection
- Mitigation and Recovery Methodologies
- Cyber Forensics and Technology to Enable Prosecution of Criminals
- Modeling and Testbeds for New Technologies
- Metrics, Benchmarks, and Best Practices
- Societal and Governance Issues


REPORT TO THE PRESIDENT
February 2005
Cyber Security: A Crisis of Prioritization
President’s Information Technology Advisory Committee

-- From Joe Jarzombek, PMP, Director for Software Assurance, NCSD, DHS

2

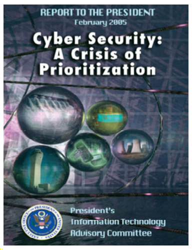
Threat

The problem has implications for:

- **Safety**
- **Privacy**
- **Security**
- **Economy**
- **Even national security**

-- From Joe Jarzombek, PMP, Director for Software Assurance, NCSD, DHS

PITAC's Findings Relative to Needs for Secure Software Engineering & Software Assurance


REPORT TO THE PRESIDENT
February 2005
Cyber Security: A Crisis of Prioritization
President's Information Technology Advisory Committee

- Commercial software engineering today lacks the scientific underpinnings and rigorous controls needed to produce high-quality, secure products at acceptable cost.
- Commonly used software engineering practices permit dangerous errors, such as improper handling of buffer overflows, which enable hundreds of attack programs to compromise millions of computers every year.
- In the future, the Nation may face even more challenging problems as adversaries – both foreign and domestic – become increasingly sophisticated in their ability to insert malicious code into critical software.

3

INFORMATION TECHNOLOGY STANDARDS

Government Response

There are initiatives underway in both DoD and DHS.

-- From Joe Jarzombek, PMP, Director for Software Assurance, NCSD, DHS

DHS Software Assurance Initiative

- **Purpose**:
  - Shift security paradigm from Patch Management to Software Assurance
  - Encourage the software developers (public and private industry) to raise the bar on software quality and security
  - Facilitate discussion, develop practical guidance, review tools, and promote R&D investment
- **Charter** -- **The National Strategy to Secure Cyberspace - Action/Recommendation 2-14:**

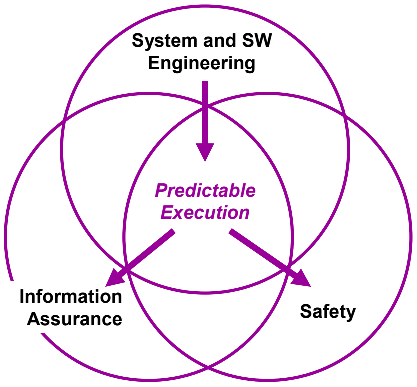
  "DHS will facilitate a national public-private effort to promulgate best practices and methodologies that promote integrity, security, and reliability in software code development, including processes and procedures that diminish the possibilities of erroneous code, malicious code, or trap doors that could be introduced during development."

7

INFORMATION TECHNOLOGY STANDARDS

Relationship of Software Assurance to Other Disciplines

Relating SW Assurance to SW Engineering



For a safety analysis to be valid …

For a security analysis to be valid …

The execution of the system must be *predictable*. This requires …

- Correct implementation of requirements, expectations and regulations. — *Traditional concern*
- Exclusion of unwanted function even in the face of attempted exploitation. — *New concern*

MITRE

Relationship of Software Assurance to Other Disciplines

Some "avoidable mistakes" are encouraged by poor usage (arguably, poor design) of programming languages.

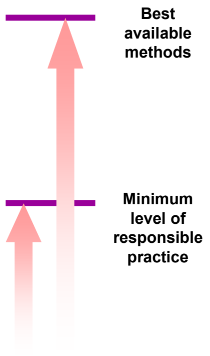
Raising the Ceiling and Raising the Floor

Raising the Ceiling

- *Information Assurance* and *System Safety* typically treat the concerns of the most critical of systems.
  - They prescribe extra practices (and possibly, extra cost) in developing, maintaining and operating such systems.

Raising the Floor

- However, *some* of the concerns of *Software Assurance* involve simple things that any developer should do.
  - They don't cost anything extra.
  - In some cases, they amount to "stop making avoidable mistakes."



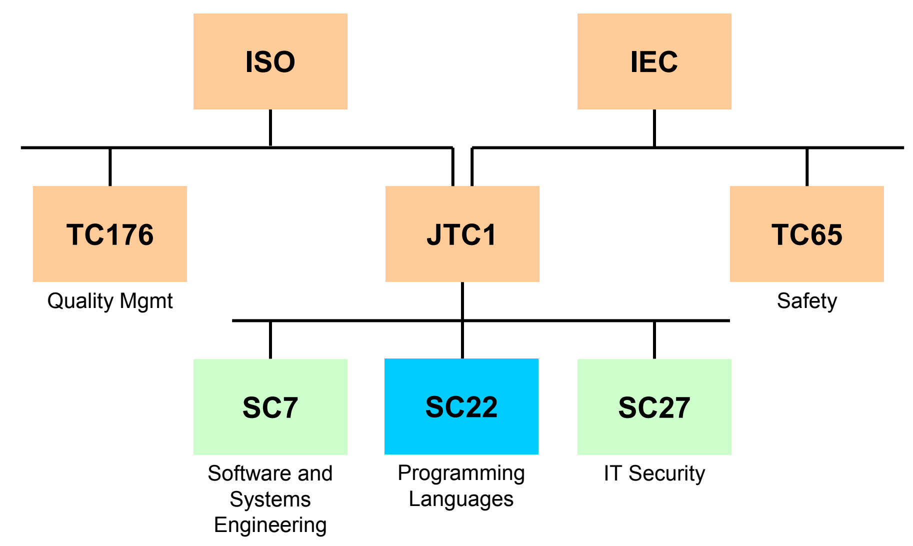
MITRE

Problem

- Any programming language has constructs that are imperfectly defined, implementation-dependent or difficult to use correctly.
- As a result, software programs sometimes execute differently than intended by the writer.
- In some cases, these vulnerabilities can be exploited by unfriendly parties.
  - Can compromise safety, security and privacy.
  - Can be used to make additional attacks.

Complicating Factors

- The choice of programming language for a project is not solely a technical decision and is not made solely by software engineers.
- Some vulnerabilities cannot be mitigated by better use of the language but require mitigation by other methods, e.g. review, static analysis.



Guidance to Avoiding Vulnerabilities in Programming Languages through Language Selection and Use (1 of 3)

- New project in ISO/IEC JTC 1/SC 22 will produce a Technical Report (which is not a "standard"). It will provide guidance, not requirements.
- Purpose: ... *prepare comparative guidance spanning a large number of programming languages, 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 ...*

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.

Guidance to Avoiding Vulnerabilities in Programming Languages through Language Selection and Use (2 of 3)

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

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’). ...

Guidance to Avoiding Vulnerabilities in Programming Languages through Language Selection and Use (3 of 3)

- *... 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 ...*

Status

- The project is now being organized.
- The project is assigned to SC22’s OWG on Vulnerability:
  - Convener, Jim Moore
  - Co-Convener, John Benito
- More information

http://aitc.aitcnet.org/isai/