The Common Weakness Enumeration (CWE) Initiative

Part of the DHS/DoD Software Assurance Initiative's Tools and Technologies Effort

CWE

cwe.mitre.org

[currently cve.mitre.org/cwe/]

June 27, 2006

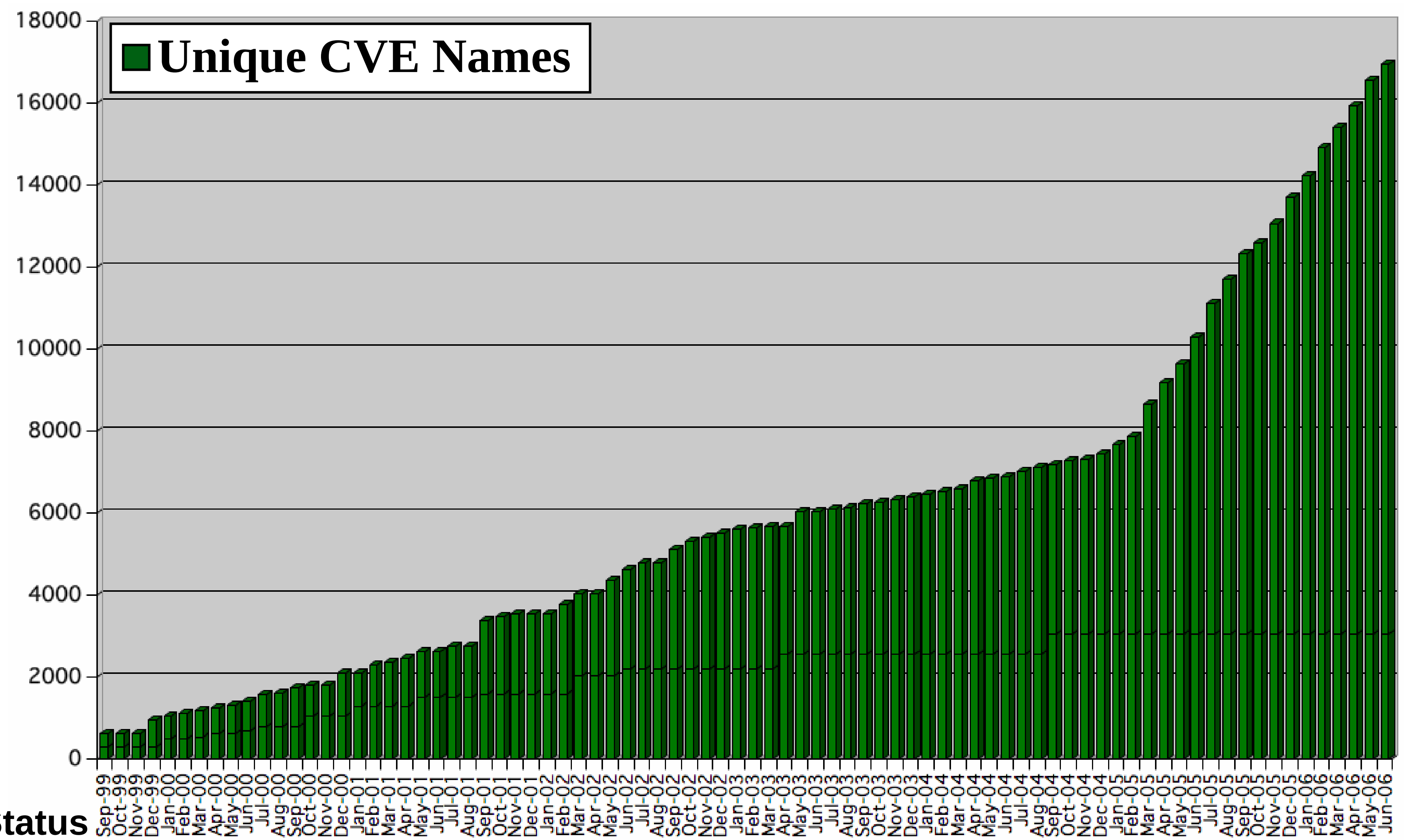
Robert A. Martin

MITRE

This Briefing Will Touch Upon Multiple Efforts Ongoing in the Software Assurance (SwA) Arena

- National Institute of Science and Technology (NIST)'s Software Assurance Metrics and Tool Evaluation (SAMATE)
- MITRE/Department of Homeland Security (DHS) Common Weakness Enumeration (CWE)
- Cigital/MITRE/DHS Common Attack Patterns Enumeration and Classification (CAPEC)
- Object Management Group (OMG) SwA Special Interest Group (SIG)

CVE Growth



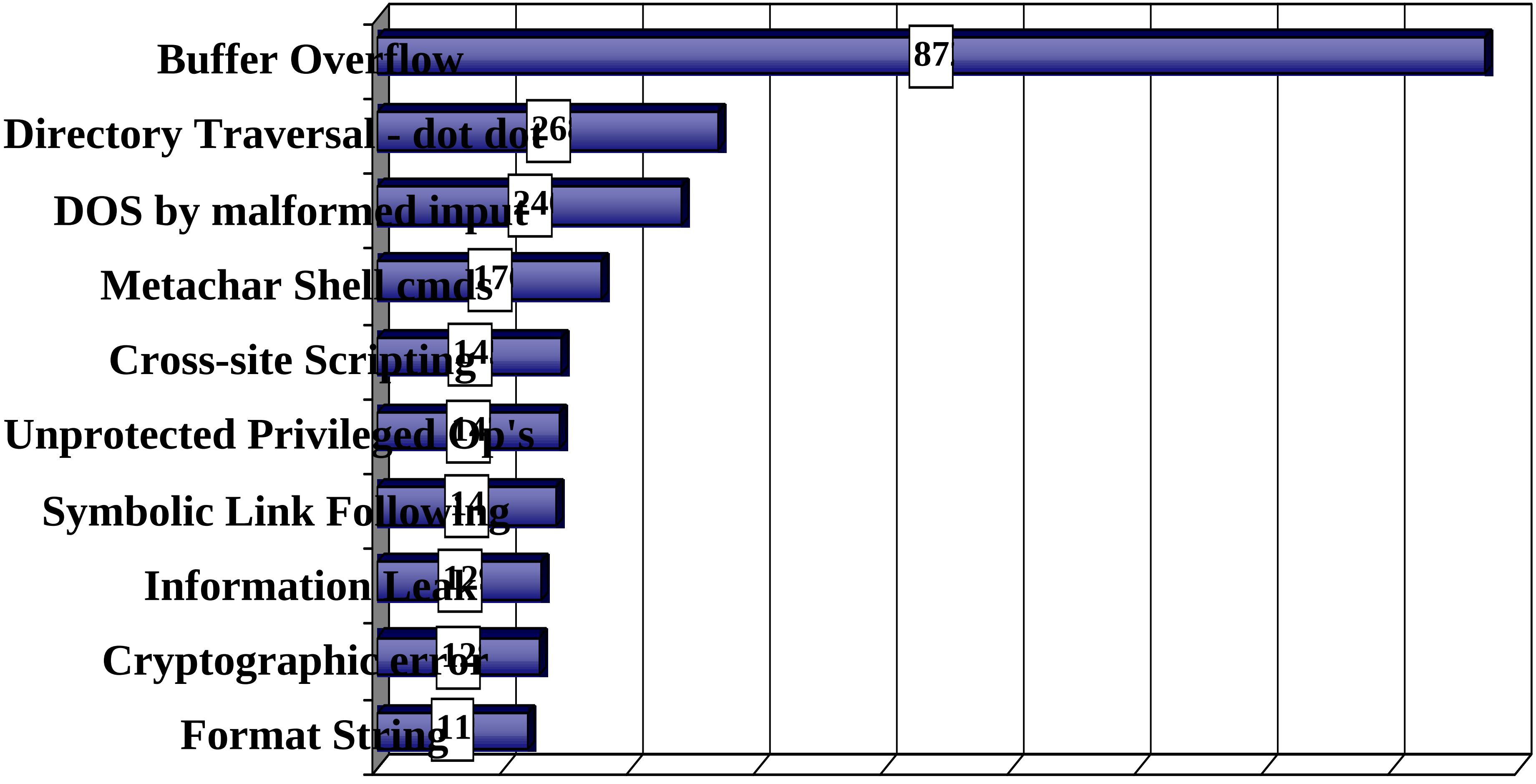
Status

(as of May 17, 2006)

- **16,943 unique CVE names**

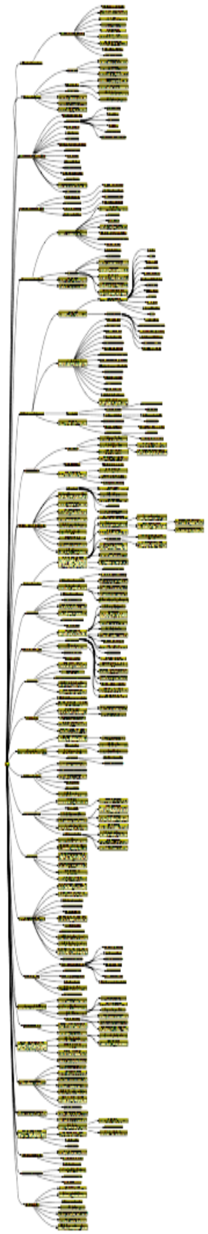
Top Ten Vulnerability Types in CVE

*(covering 2361 of 3933 *CVE issues publicized between 1 Jan 2000-13 Feb 2003 inclusive)*



**** The "Types" of Other and Unknown represent 831vulnerabilities***

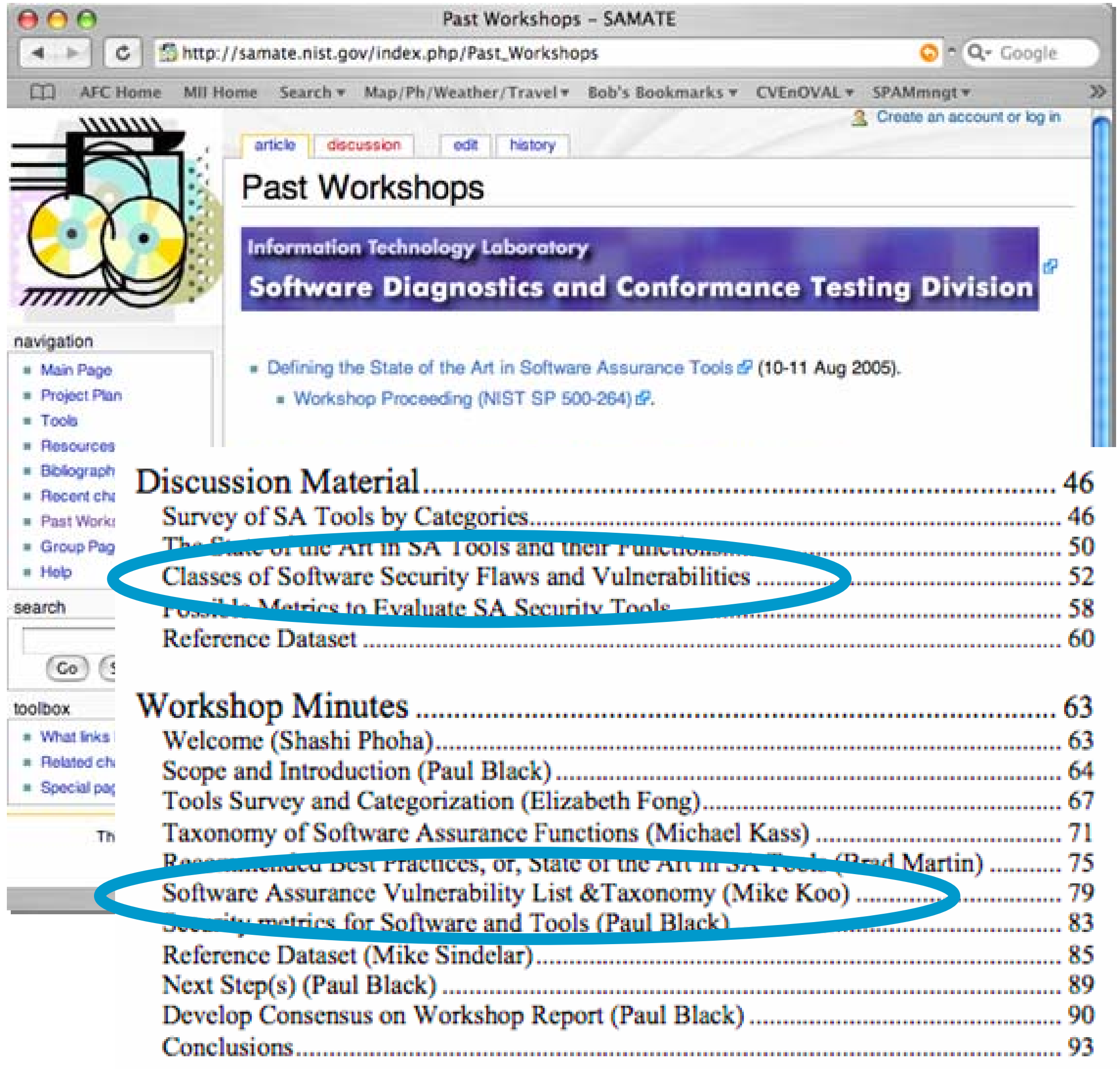
Preliminary List of Vulnerability Examples for Researchers (PLOVER)
300 “types”of Weaknesses



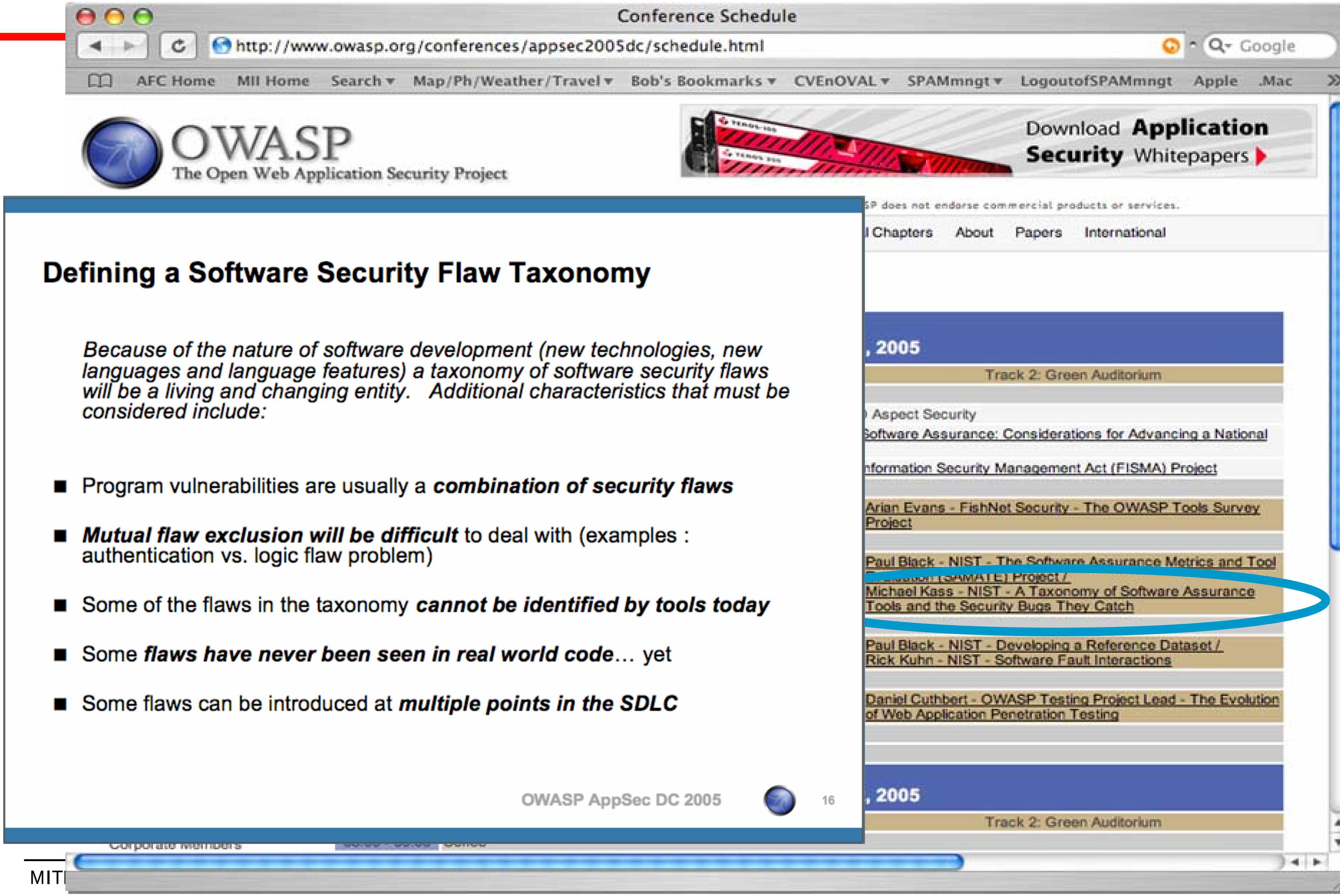
[BUFF] Buffer overflows, format strings, etc.	10 types
[SVM] Structure and Validity Problems	10 types
[SPEC] Special Elements (Characters or Reserved Words)	19 types
[SPECM] Common Special Element Manipulations	11 types
[SPECTS] Technology-Specific Special Elements	17 types
[PATH] Pathname Traversal and Equivalence Errors	47 types
[CP] Channel and Path Errors	13 types
[CCC] Cleansing, Canonicalization, and Comparison Errors	16 types
[INFO] Information Management Errors	19 types
[RACE] Race Conditions	6 types
[PPA] Permissions, Privileges, and ACLs	20 types
[HAND] Handler Errors	4 types
[UI] User Interface Errors	7 types
[INT] Interaction Errors	7 types
[INIT] Initialization and Cleanup Errors	6 types
[RES] Resource Management Errors	11 types
[NUM] Numeric Errors	6 types
[AUTHENT] Authentication Error	12 types
[CRYPTO] Cryptographic errors	13 types
[RAND] Randomness and Predictability	9 types
[CODE] Code Evaluation and Injection	4 types
[ERS] Error Conditions, Return Values, Status Codes	4 types
[VER] Insufficient Verification of Data	7 types
[MAID] Modification of Assumed-Immutable Data	2 types
[MAL] Product-Embedded Malicious Code	7 types
[ATTMIT] Common Attack Mitigation Failures	3 types
[CONT] Containment errors (container errors)	3 types
[MISC] Miscellaneous WIFFs	7 types

Flaw Taxonomy Discussions Started as part of the NIST SAMATE Effort:

- Need for Flaw Taxonomy Identified as Supporting activity to NIST SAMATE effort to measure effectiveness of tools in finding these weaknesses



The Open Web Application Security Project Conference (11-12 Oct 2005)



SSATTM – SAMATE

http://samate.nist.gov/index.php/SSATTM

AFC Home | MII Home | Search | Map/Ph/Weather/Travel | Bob's Bookmarks | CVEnOVAL | SPAMmngt

Create an account or log in

article | discussion | edit | history

navigation

- Main Page
- Project Plan
- Tools
- Resources
- Bibliography
- Recent changes
- Past Workshops
- Group Pages
- Help

search

Go | Search

toolbox

- What links here
- Related changes
- Special pages

SSATTM

Information Technology Laboratory

Software Diagnostics and Conformance Testing Division

Workshop on Software Security Assurance Tools, Techniques, and Metrics

PROGRAM

[edit]

November 7, 2005

8:30 – 9:00 : Welcome – Paul E. Black

9:00 – 10:30 : Tools and Metrics - Liz Fong

- Where do Software Security Assurance Tools Add Value? – David Jackson, David Cooper
- Metrics that Matter – Brian Chess
- The Case for Common Flaw Enumeration – Robert Martin, Steven Christey, Joe Jarzombek

10:30 – 11:00 : Break

11:00 – 12:30 : Flaw Taxonomy and Benchmarks - Robert Martin

- Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors – Katrina Tsipenyuk, Brian Chess, Gary McGraw
- A Taxonomy of Buffer Overflows for Evaluating Static and Dynamic Software Testing Tools – Kendra Kratkiewicz, Richard Lippmann
- ABM – A Prototype for Benchmarking Source Code Analyzers – Tim Newsham, Brian Chess

Goal for the Common Weakness Enumeration:

- To improve the quality of software with respect to known security issues within source code
  - define a unified measurable set of weaknesses
  - enable more effective discussion, description, selection and use of software security tools and services that can find these weaknesses

What does CWE need to come to agreement on?

- Two separate but synergistic sub-goals:
  - Need detailed and specific definitions of the individual issues that we want to remove/reduce in software (a dictionary)
  - Need a structure/organization for thinking about the issues and allowing discussion/debate about entire groupings of issues (views/taxonomies)

Clarifying software weaknesses: Enabling communication (1 of 2)

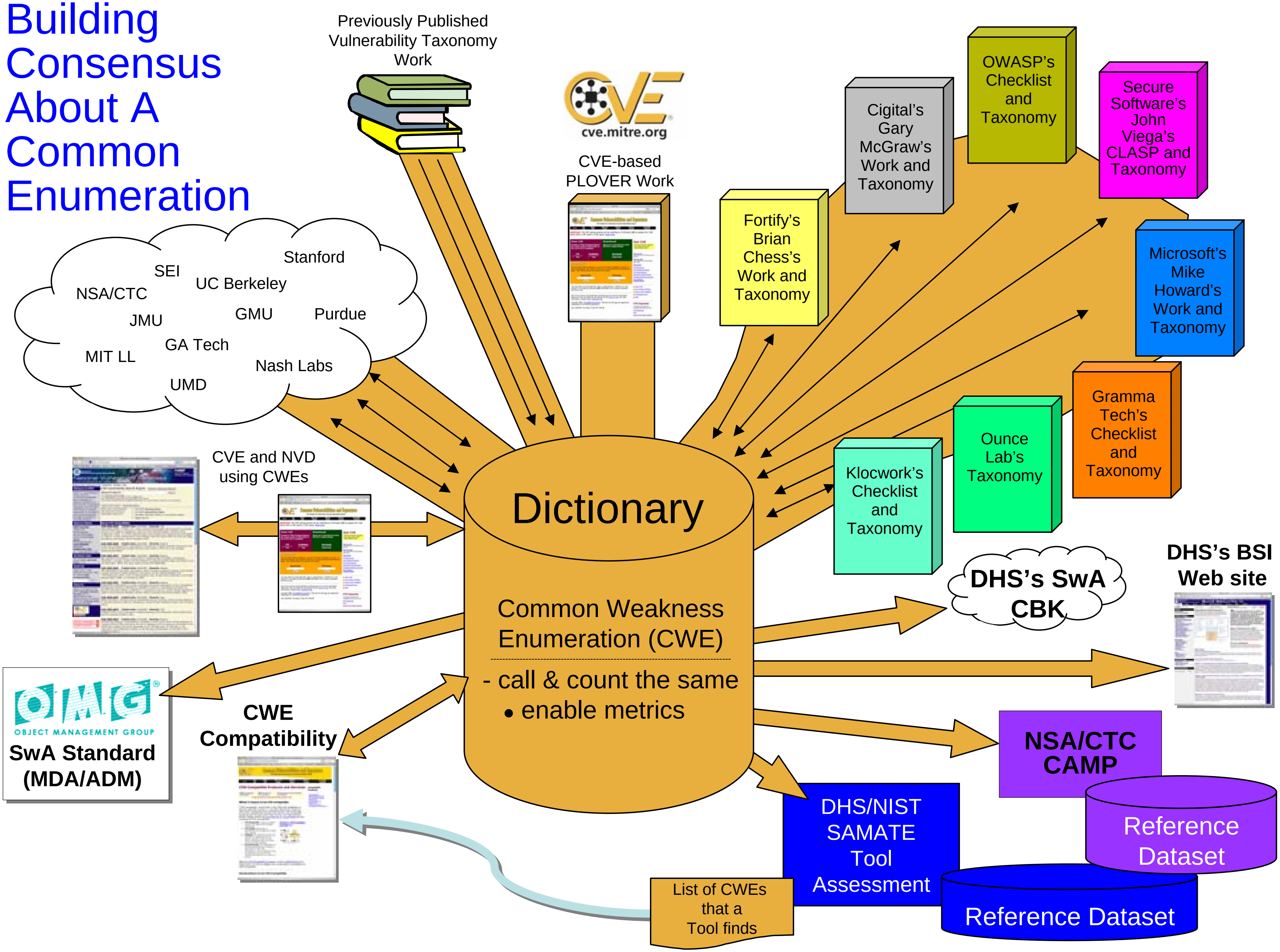
- Systems Development Manager Issue Areas:
  - What are the software weaknesses I need to protect against
    - Architecture, design, code
  - Can I look through the issues by technologies, risks, severity
  - What have the pieces of my system been vetted for?
    - COTS packages, organic development, open source
  - Identify tools to vet code based on tool coverage
    - How effective are the tools?

- Assessment Tool Vendors Issue Areas:
  - Express what my tool does
  - Succinctly identify areas I should expand coverage

Clarifying software weaknesses: Enabling communication (2 of 2)

- COTS Product Vendor Issue Areas:
  - What have I vetted my applications for?
  - What do my customers want me to vet for?
- Researcher Issue Areas:
  - Quickly understand what is known
  - Easily identify areas to contribute/refine/correct
- Educator Issue Areas:
  - Train students with the same concepts they'll use in practice
- Operations Manager Issue Areas:
  - What issues have my applications been vetted for? (COTS/Organic/OS)
  - What types of issues are more critical for my technology?
  - What types of issues are more likely to be successfully exploited?

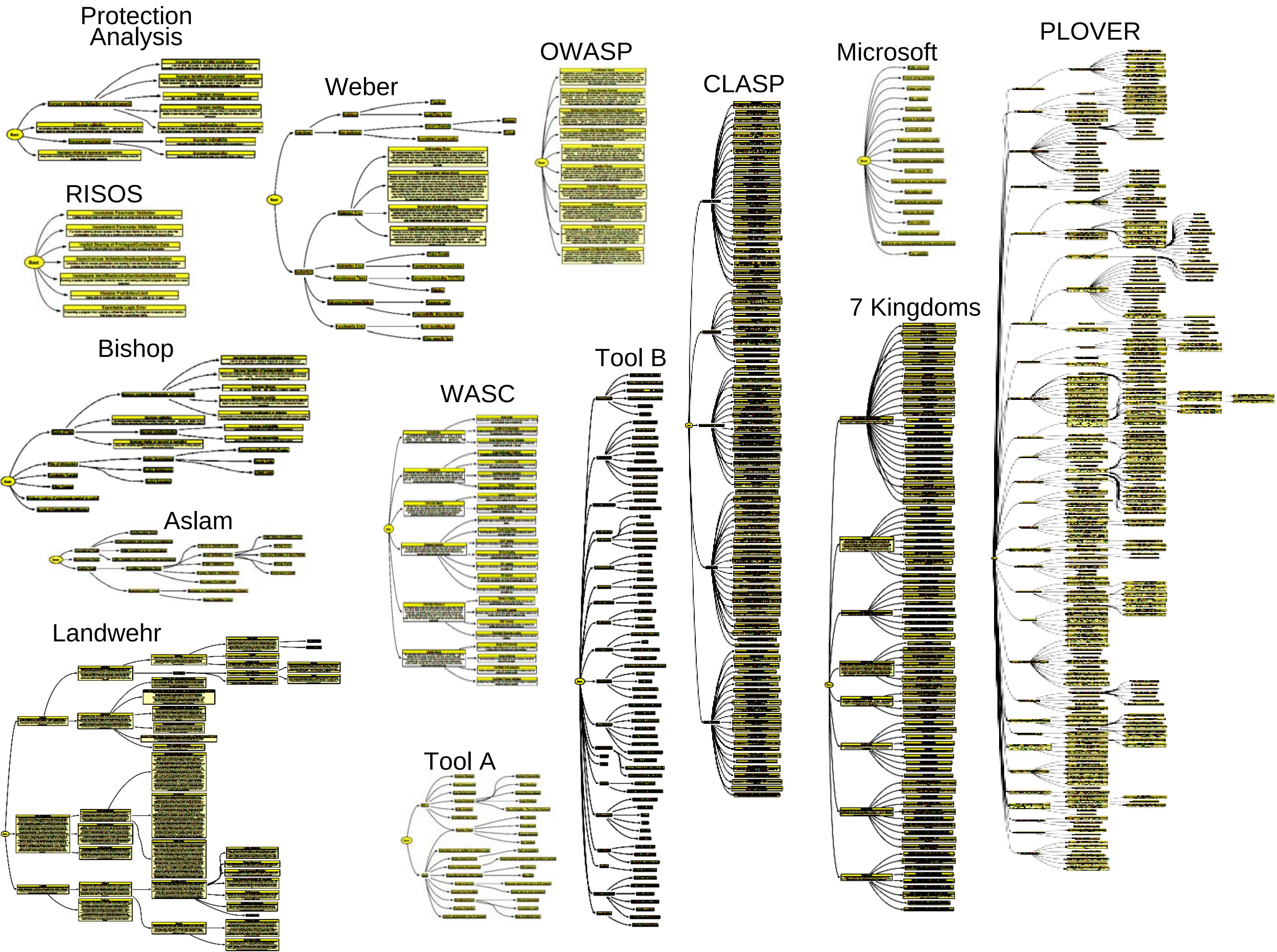
Building Consensus About A Common Enumeration



SwA Metrics & Tool Evaluation (SAMATE)

* SAMATE Reference Dataset (SRD), version 2, on-line
  This dataset will have 1000s of test cases for evaluation and development of SwA tools. Cases will have breadth of
  - language (C, Java, UML, etc.)
  - life cycle (design model, source code, application, …)
  - size and type (small and huge, production and artificial, …)
* Specifications and a reviewed test, including a suite of test cases (from the SRD above) for one class of SwA tool, probably source code scanners.
* Specifications & test for another class of SwA tool, probably web applications.
* Establish an advisory committee and create a road map to creating tests for all SwA tools (which tool classes should be done first?).
* List SwA areas with underdeveloped tools; sketch R&D that could fill each area.
* Requires Common Enumeration of Weaknesses (CWE) to provide a dictionary of software flaws

SAMATE project leader, Paul E. Black, paul.black@nist.gov (p.black@acm.org),
100 Bureau Drive, Stop 8970, Gaithersburg, Maryland 20899-8970
voice: +1 301 975-4794, fax: +1 301 926-3696, http://hissa.nist.gov/~black/ KC7PKT


Protection Analysis
RISOS
Bishop
Aslam
Landwehr
Weber
WASC
Tool A
OWASP
Tool B
CLASP
Microsoft
7 Kingdoms
PLOVER

Current Community Contributing to the Common Flaw Enumeration

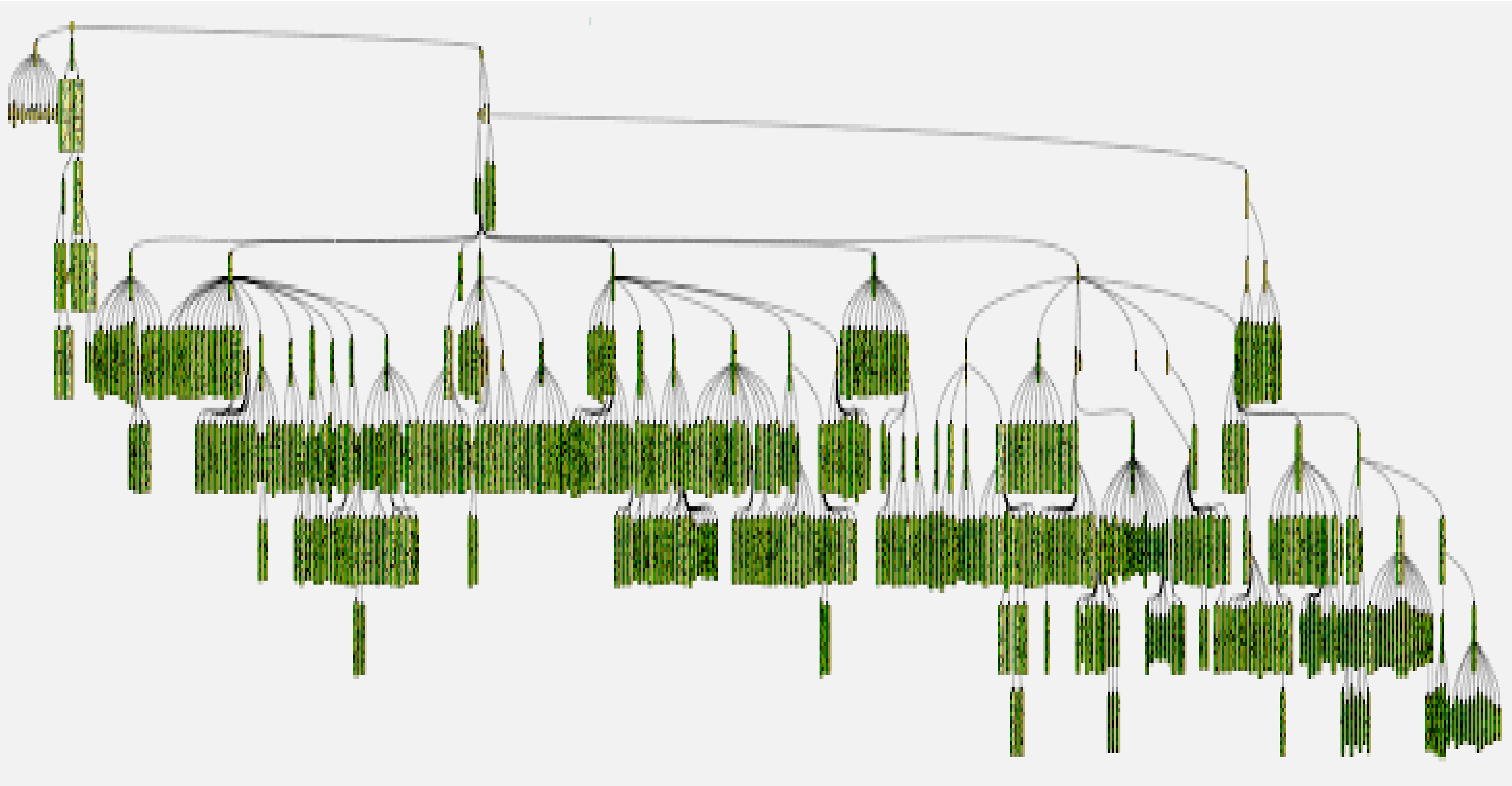
- Cenzic
- CERIAS/Purdue University
- CERT/CC
- Cigital
- CodescanLabs
- Core Security
- Coverity
- DHS
- Fortify
- IBM
- Interoperability Clearing House
- JHU/APL
- Kestrel Technology
- KDM Analytics
- Klocwork
- Microsoft
- MIT Lincoln Labs
- MITRE
- North Carolina State University
- NIST
- NSA
- Oracle
- Ounce Labs
- OWASP
- Parasoft
- proServices Corporation
- Secure Software
- Security University
- Semantic Designs
- SPI Dynamics
- UNISYS
- VERACODE
- Watchfire
- WASC
- Whitehat Security, Inc.

- Tim Newsham

To join send e-mail to cwe@mitre.org

CWE 2nd Draft is available @ [cve.mitre.org/cwe]

Approximately 500 Dictionary Elements



Approximately 500 Dictionary Elements

The Classification Tree Nodes Link to the Dictionary Entries

CWE Classification Tree (initial draft)

- Common Weakness Enumeration and Classification
  - Location
    - Environment
    - Configuration
    - Code
      - Source Code
        - Data Handling
          - Input Validation
          - Output Validation
          - Range Errors
            - Buffer Errors
              - OVER - Unbounded Transfer ('classic overflow')
                - Stack overflow
                - Heap overflow
              - Write-what-where condition
              - UNDER - Boundary beginning violation ('buffer underwrite')
              - READ - Out-of-bounds Read
              - Wrap-around error
              - Unchecked array indexing
              - LEN - Length Parameter Inconsistency
              - LENCALC - Other length calculation error
              - Miscalculated null termination
              - String Errors
          - Type Errors
          - Representation Errors

BACK TO TOP

CWE Dictionary (initial draft)

() | 1-9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

Stack overflow	
Description	A stack overflow condition is a buffer overflow condition, where the buffer being overwritten is allocated on the stack (i.e., is a local variable or, rarely, a parameter to a function).
Likelihood of Exploit	Very high
Weakness Ordinality	Primary (Weakness exists independent of other Weaknesses)
Causal Nature	Explicit (This is an explicit weakness resulting from behavior of the developer)
Common Consequences	Availability: Buffer overflows generally lead to crashes. Other attacks leading to lack of availability are possible, including putting the program into an infinite loop. Access control (memory and instruction processing): Buffer overflows often can be used to execute arbitrary code, which is usually outside the scope of a program*s implicit security policy. Other: When the consequence is arbitrary code execution, this can often be used to subvert any other security service.
Potential Mitigations	Pre-design: Use a language or compiler that performs automatic bounds checking. Design: Use an abstraction library to abstract away risky APIs. Not a complete solution. Pre-design through Build: Compiler-based canary mechanisms such as StackGuard, ProPolice and the Microsoft Visual Studio /GS flag. Unless this provides automatic bounds checking, it is not a complete solution. Operational: Use OS-level preventative functionality. Not a complete solution.

What would be the details of the definitions?

- Name for an issue type
- Description of the type
- Description of the behavior of the issue
- Description of the exploit of the issue
- Description of the impact of the exploit
- Code samples for the languages/architectures where the issue exists
- CVE names of vulnerabilities of that issue type
- ...?

Currently CWE has:

- Name, Description, Alternate Terms, Likelihood of Exploit, Weaknes Ordinality, Causal Nature, Common Consequences, Potential Mitigations, Observed Examples, Context Notes, References, Node Relationships, and Source Taxonomies

Using A Unilateral NDA with MITRE to Bring in Info

Purpose:

- Sharing the proprietary/company confidential information contained in the underlying Knowledge Repository of the Knowledge Owner's Capability for the sole purpose of establishing a public Common Weakness Enumeration (CWE) dictionary that can be used by vendors, customers, and researchers to describe software, design, and architecture related weaknesses that have security ramifications.

- The individual contributions from numerous organizations, based on their proprietary/company-confidential information, will be combined into a consolidated collection of weakness descriptions and definitions with the resultant collection being shared publicly.

- The consolidated collection of knowledge about weaknesses in software, design, and architecture will make no reference to the source of the information used to describe, define, and explain the individual weaknesses.

FORTIFY Klocwork CORE Security VERACODE
CENZIC WatchFire Secure Software
proServices Coverity SPI DYNAMICS OUNCE LABS Parasoft

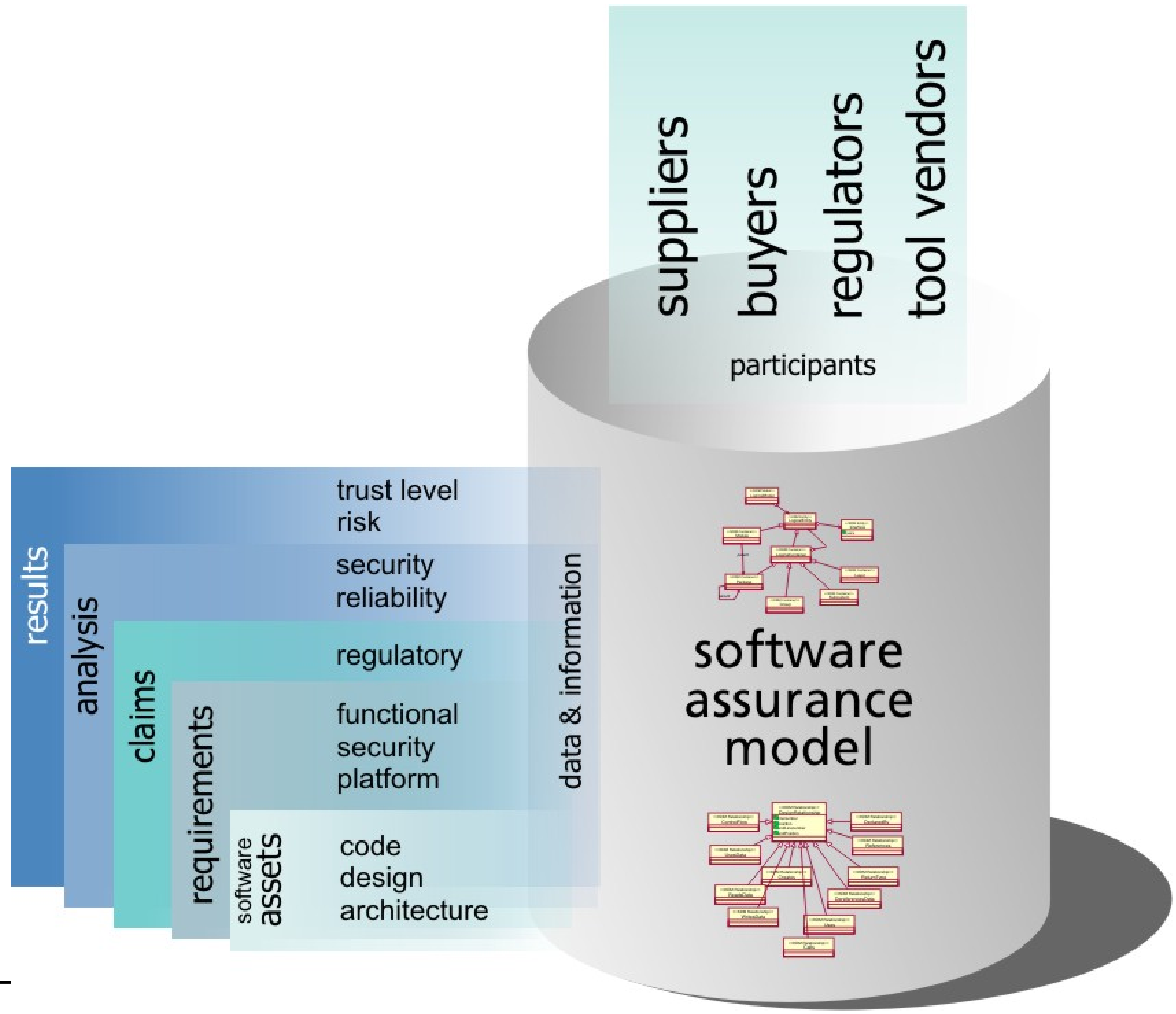
Common Attack Patterns Enumeration and Classification (CAPEC)

- **Description**
  - Supports classification taxonomies to be easily understood and consumable by the broad software assurance community and to be aligned and integrated with the other SwA community knowledge catalogs.
- **Tasks**
  - Identify and analyze reference Attack Pattern resources from academia, govnt, and industry.
  - Define standard Attack Pattern schema.
  - Identify and collect potential Attack Pattern seedling instances.
  - Finalize scope of effort to clarify number of Attack Patterns to be targeted for initial release.
  - Translate Attack Pattern seedling content into the defined schema.
  - Analyze and extend Attack Pattern seedlings to fulfill schema.
  - Identify set of new Attack Patterns to be authored.
  - Author targeted list of new Attack Patterns.
  - Map all Attack Patterns to the Common Weaknesses Enumeration (CWE).
  - Define a classification taxonomy for Attack Patterns.
  - Map Attack Patterns into the defined classification taxonomy.
  - Publish content to SwA community, solicit input, collaborate, review, and revise as needed.
  - Define process for ongoing extension and sustainment of the CAPEC.
  - Provide assistance to design, build, test, and deploy a website for public hosting of CAPEC.

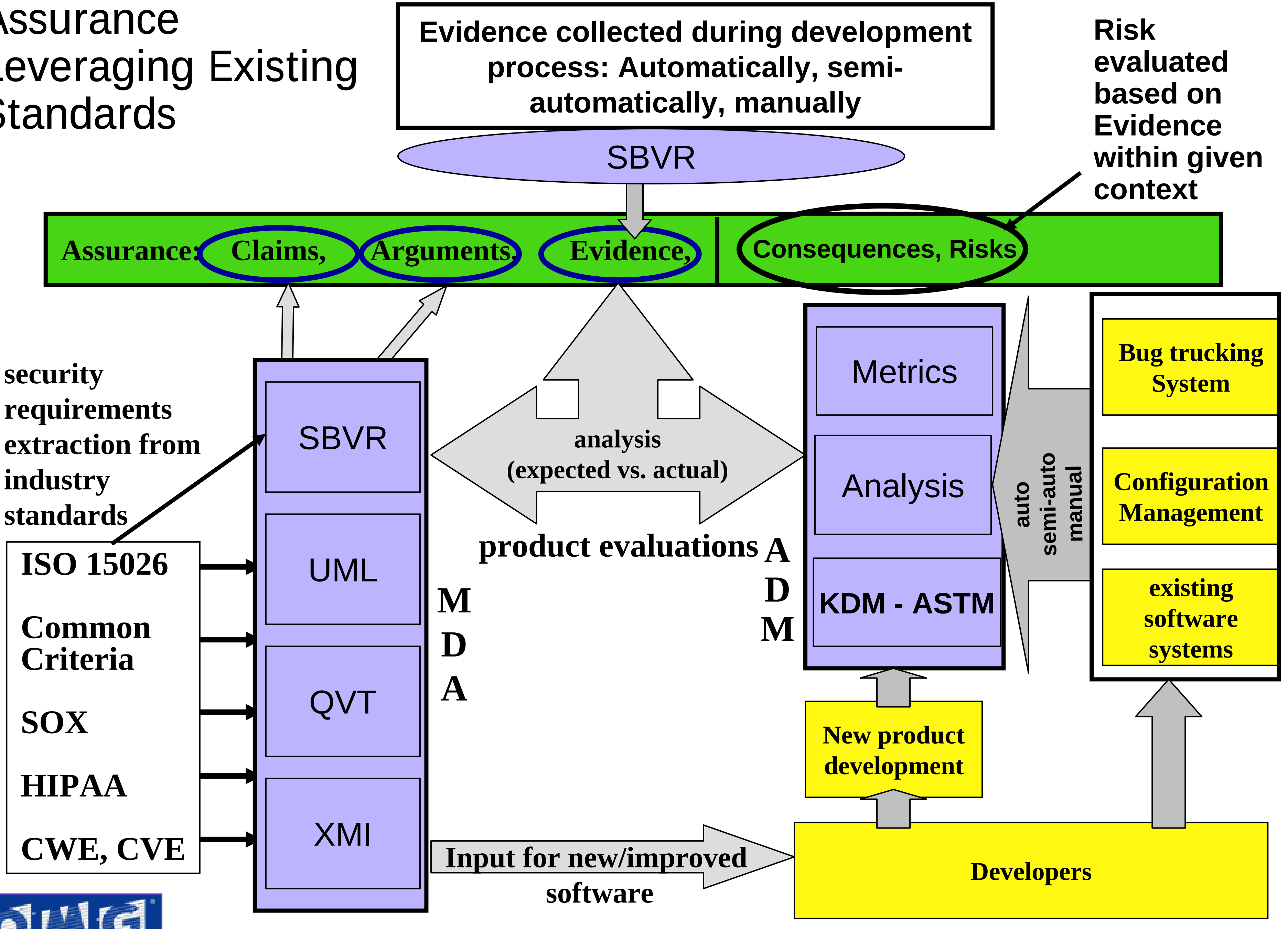
The Challenge for the OMG SwA SIG:
How Do You Measure an Abstract Concept Like Secureness?

Standardization will ensure that all participants are investing not just in individual activities but in a coordinated strategy.

- Software Assurance (SwA) standardization will establish interoperability for exchange of information among participants
- Standardization of SwA means …
  - A formalization of the set of common definitions related to SwA
  - A format for exchanging information related to SwA



Assurance Leveraging Existing Standards



The Road Ahead for the CWE effort

- Finishing the strawman dictionary/ taxonomy DONE
- Creating a web presence DONE
- Getting NDAs with knowledgeable organizations DONE
- Getting agreement on the detailed enumeration
- Dovetailing with test cases (NIST/CAMP)
- Dovetailing with attack patterns (Cigital)
- Dovetailing with coding standards (SEI CERT/CC)
- Dovetailing with BSI, CBK, OMG SwA SIG, SC22,…
- Create alternate views into the CWE dictionary

Acronyms from this Presentation

ADM	Architecture Driven Modernization
BSI	Build Security In
CBK	Common Body of Knowledge
CVE	Common Vulnerabilities and Exposures
CWE	Common Weakness Enumeration
COTS	Commercial Off The Shelf
DHS	Department of Homeland Security
DITSCAP	DoD Information Technology Security Certification & Accreditation Process
DoD	Department of Defense
FISMA	Federal Information Security Management Act
HIPPA	Health Insurance Portability and Accountability Act
KDM	Knowledge Discovery Meta-Model
MDA	Model Driven Architecture
NDA	Non Disclosure Agreement
NIST	National Institute of Science and Technology
NSA	National Security Agency
NVD	National Vulnerability Database
OMG	Open Management Group
OWASP	Open Web Application Security Project
PLOVER	Preliminary List of Vulnerability Examples for Researchers
SAMATE	Software Assurance Measurement and Tool Evaluation
SIG	Special Interest Group
SOX	Sarbanes-Oxley
SPEM	Software Process Engineering Metamodel
SSATTM	Software Security Assurance Tools, Techniques, and Metrics
SwA	Software Assurance
XML	Extensible Markup Language