
Certifying Applications for Known Security Weaknesses

The Common Weakness Enumeration (CWE) Effort

Robert A. Martin - MITRE

6 March 2007

Software Security Assurance

Don't Let Your Applications Get You Down

SECURE

Exploited software flaws cost the U.S. financial services industry more than \$3 billion per year, according to the National Institute of Standards & Technology.

BY JOHN K. WATERS

Most enterprises have figured out that antivirus software and intrusion detectors, although essential to their security posture, no longer provide adequate protection. Black-hat hackers and other intruders are increasingly circumventing network security and exploiting loopholes in applications.

These vulnerabilities are costing enterprises a National Institute of Standards & Technology companies are spending about \$60 billion a year and costing the U.S. financial services industry more than \$3 billion, according to NIST.

Part of the problem is the changing nature of enterprise IT. Data centers and access are no longer exclusively by apps within the enterprise's security perimeter, but are now connected to customers, partners and suppliers through the Web.

Another part of the problem is the nature of application development, he says. Application development is now rewarded for delivering features and meeting customer needs, not for focusing on those things. Consequently, security becomes an afterthought, and developers focus on those things only if they are discovered vulnerabilities have been developed.

Not doing the security work up front can get expensive, Gartner figures it's 10 times more cost

The DoD SoftwareTech NEWS

Secure Software Engineering

The Challenge of Low Defect, Secure Software

| | |
|-------------------------------|----|
| Enhancing Customer Security | 71 |
| Software Development Security | 75 |
| User Comment | 78 |

Procurement is key to security, IT execs say

BY PATIENCE WAT | GSN STAFF

Procurement officers have the power to significantly improve the security of government IT systems by including software reliability and security requirements in the contracts they award to vendors—and

INFORMATION ASSURANCE

strengthen the country's cyber-infrastructure in the process.

That key message was hammered home repeatedly at a two-day forum earlier this month hosted jointly by the Defense and Homeland Security departments.

"We have to shift the paradigm from patch management to software assurance," said Andy Prady, acting director of DHS' National Cyber Security Division.

Vendors will not invest in improving the quality of their software of their own volition, said Patricia Getchis, deputy CIO and deputy assistant secretary of Defense for networks and information integration.

"We've got to use acquisition organiza-

tion organizations to put together a software assurance policy. We have to ... make sure [it's] part of the way we buy."

—BOB PRINCE, DHS

Update gives developers head start in fixing code

By Paul F. Roberts

A NEW SOFTWARE RELEASE from Secure Software Inc. is designed to make it easier for organizations doing software development to spot and resolve security flaws in raw computer code.

Secure Software, based in McLean, Va., planned to release in late June CodeAssure 2.0, the latest edition of its automated code security auditing technology. The upgrade includes CodeAssure Management Center, a tool that will make it easier to

Red Hat Enterprise Linux and Novell Inc.'s SUSE Linux and some versions of Unix, Kernan said.

The new Management Center component lets managers track vulnerability trends, prioritize code fixes, set and enforce policies for fixing vulnerabilities, monitor the status of code review projects, and create reports and business impact assessments of individual projects or project portfolios.

CodeAssure can be used as a plug-in with the Eclipse



open-source IDE (integrated development environment) from the Eclipse Foundation Inc.

Microsoft Corp. said in June that it was working with SPI Dynamics to integrate its DesInspect and SecureObjects into Visual Studio 2005 and Visual Studio 2005



Nominations are now open, click here to

Go to - SC Magazine Asia Pacific

Latest News

Software quality, FISMA top federal CISO concerns

by Marcia Savage
[Mon, Aug 29, 2005] Software quality and FISMA compliance topped a list of concerns expressed by federal CISOs in a recent survey.

Conducted by Intelligent Decisions, a Washington, D.C.-based systems integrator, the survey of 29 federal CISOs ranked increased software quality assurance as the top area that the private sector needs to focus on.

Software Assurance



OSD

Background

- In October 2002, the President's Critical Infrastructure Protection Board (PCIPB) created the National Security Agency (NSA) -led IT Security Study Group (ITSSG) to review existing IT acquisition processes.
- In July 2003, the Assistant Secretary of Defense for Networks and Information Integration (ASD(NII)) established the Software Assurance Initiative to examine software assurance issues
- On 23 Dec 04, Undersecretary of Defense for Acquisitions, Technology and Logistics (USD(AT&L)) and ASD(NII) established a Software Assurance (SwA) Tiger Team to:
 - Develop a holistic strategy to reduce SwA risks within 90 days
 - Provide a comprehensive briefing of findings, strategy and plan
- Tiger Team presented its strategy to USD(AT&L) and ASD(NII) on March 28, and on May 2 was tasked to proceed with 180 day Implementation Phase

Software Assurance (SwA) Definition

Software assurance (SwA) is the level of confidence that software is free of exploitable vulnerabilities, either intentionally or unintentionally designed as part of the software or inadvertently created.

Requirements

- What functional statements in OSD Guidance for SwA requirements best enable optimal vendor solutions?
 - Require higher level written policy to specify need for SwA requirements
 - "Compelling arguments and evidence that...commensurate with risk"
 - Written SwA Principles in policy
 - Looked at 8500, 9000.2, 5000, 3170, 6212, ...
 - In 8500.2 Annex language to potentially leverage for SwA:
 - "...use IA best practices..."
 - "...software will be well behaved..."
 - Point to language in contracts
 - Contract language to show equivalence to ISO 15026 practices
 - Burden on PMO to understand and have confidence in level of SwA
 - Requirement in policy that whenever a new risk is ID's or an old risk changes, contractor must be notified



Basis for SwA Technology

- Offensive side
 - Pedigree problem
 - Intelligence
 - Value
 - Market
- Defensive side
 - Fragmented effort
 - Immature science
 - Lack of resources

NSA

Software Assurance: A Strategic Initiative of the U.S. Department of Homeland Security to Promote Integrity, Security, and Reliability in Software

Consistent with the National Security Strategy for America's Cyber Security

Sept 2004

Homeland Security

Directorate for Software Assurance
National Cyber Security Division
U.S. Department of Homeland Security

DHS

Driving Needs for Software Assurance

- Software vulnerabilities jeopardize intellectual property, business operations and services, infrastructure operations, and consumer
- Growing awareness and concern over the ability of an adversary to subvert the software supply chain
 - Federal Government relies on COTS products and commercial devices from foreign and non-vetted domestic suppliers to meet majority of IT requirements
 - Software development offers opportunities to insert malicious code or design and build software enabling exploitation
- Growing concern about inadequacies of suppliers' capabilities and deliver secure software with requisite levels of integrity
 - Current education & training provides too few practitioners with requisite competencies in secure software engineering
 - Concern about suppliers not exercising minimum level of responsibility
 - Growing need to improve both the state-of-the-practice and the state-of-the-art in software capabilities of the nation
- Processes and technologies are required to build trust into software acquired and used by Government and critical infrastructure

Strengthen operational resiliency

NIST

Motivation for Classes of Software Security Flaws & Vulnerabilities

- *For Systematic Study* – classify security problems in software into categories that one can dissect for systematic study.
- *For SS Tools Evaluation*- a taxonomy of security vulnerability that the SA community would agree upon will be essential for evaluating Software Security (SS) tools and classifying SA functions.
- *For SRD Development* - Classes of software security flaws and vulnerabilities is one of resources to drive a standard reference dataset, which, in simply put, is a benchmark test suite for Software Security tools.

NIST SAMATE Workshop: Defining the State of the Art in Software Assurance Tools (10-11 Aug 2005)

Possible Goals of Classifying Software Security Flaws & Vulnerabilities

- A taxonomy that has classification categories with the satisfactory characteristics as possible.
- Incorporate commonly used terms in security vulnerabilities that occurred in modern days.
- Consensus from the SA community.

[article](#)
[discussion](#)
[edit](#)
[history](#)

SSATTM

Information Technology Laboratory

Software Diagnostics and Conformance Testing Division

Workshop on Software Security Assurance Tools, Techniques, and Metrics

[\[edit\]](#)

PROGRAM

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

SE



navigation

- [Main Page](#)
- [Project Plan](#)
- [Tools](#)
- [Resources](#)
- [Bibliography](#)
- [Recent changes](#)
- [Past Workshops](#)
- [Group Pages](#)
- [Help](#)

search

toolbox

- [What links here](#)
- [Related changes](#)
- [Special pages](#)



Goal of the Common Weakness Enumeration Initiative

- 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

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?

CWE Launched March 2006 with draft 1, now at draft 5

CWE - Common Weakness Enumeration
A community-developed dictionary of common software weaknesses

[View the CWE List](#)

CWE List
Full Dictionary View
Classification Tree
Other Views

About
Sources
Process
Documents

Community
Related Activities

News
Calendar

Compatibility
Program
Requirements
Make a Declaration

Contact Us
Search the Site

International in scope and free for public use, CWE™ provides a unified, measurable set of software weaknesses that will enable more effective discussion, description, selection, and use of software security tools and services that can find these weaknesses in source code.

Building CWE & Consensus

News

- [CWE Draft 5 Released](#)
- [Important Message about CWE Web Site Availability, January 13-16](#)
- ["CWE Compatibility and Effectiveness" Section Added to CWE Web Site](#)
- [CWE Information Included in Article in SC Magazine](#)
- [Vulnerability Type Distributions White Paper Now Available](#)

[...more](#)

Upcoming Event

- [CWE booth at RSA 2007, February 3-11, 2007](#)
- [CWE booth at 2007 JA Workshop, February 12-16, 2007](#)

[...more](#)

Status Report

Recent

Fifth draft posted December 15, 2006. Changes include (1) additional descriptions and mitigations for about 40 of the items, (2) minor revisions and updates to approximately 100 items based on the donated information, and (3) revisions to the names and structure of the hierarchical view to reflect the new and revised CWE content.

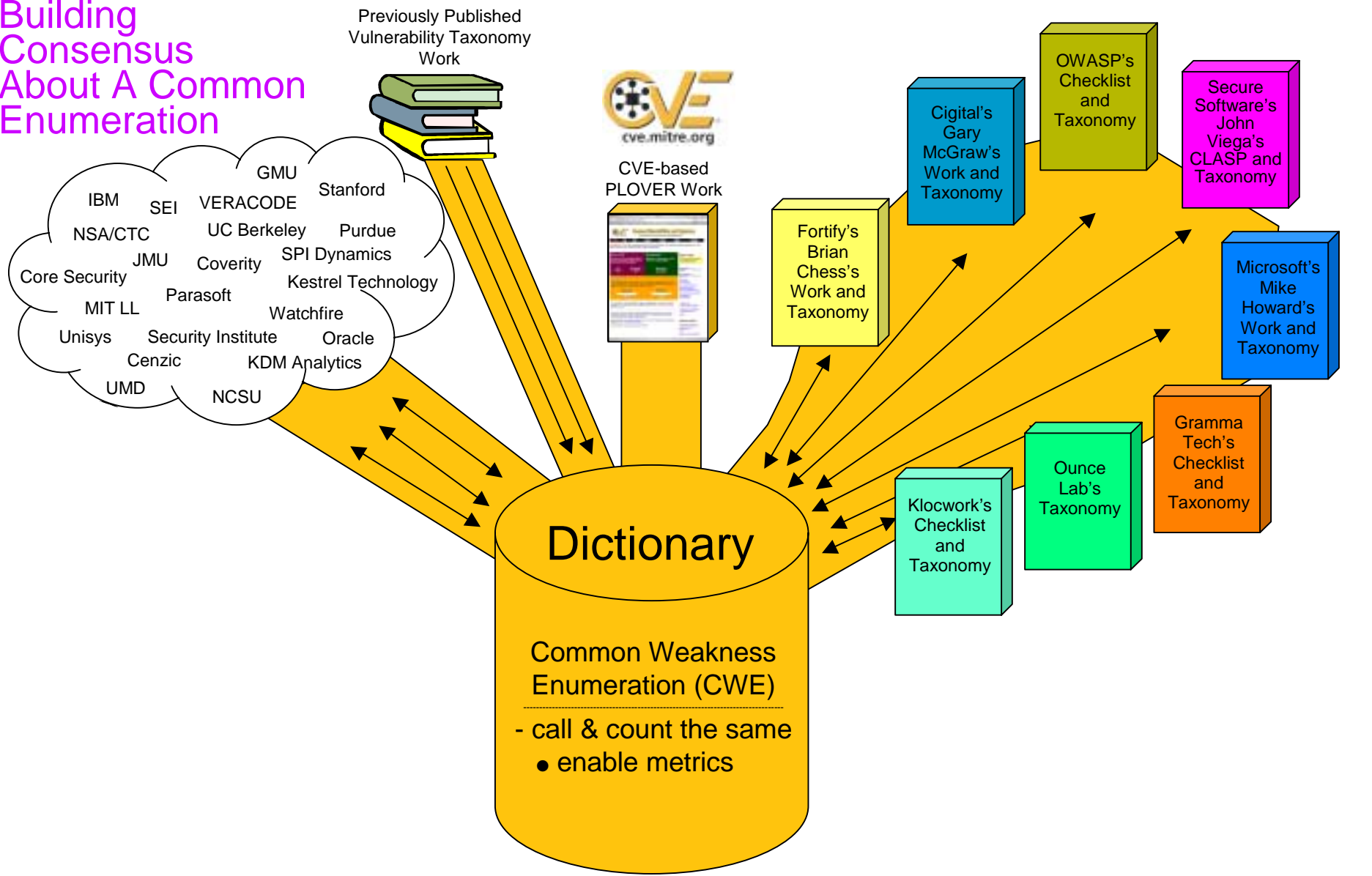
Next Step

We are gathering data about weaknesses from the sixteen tool and knowledge sources that are participating in CWE and then merging this new data into the current list to create a sixth draft.

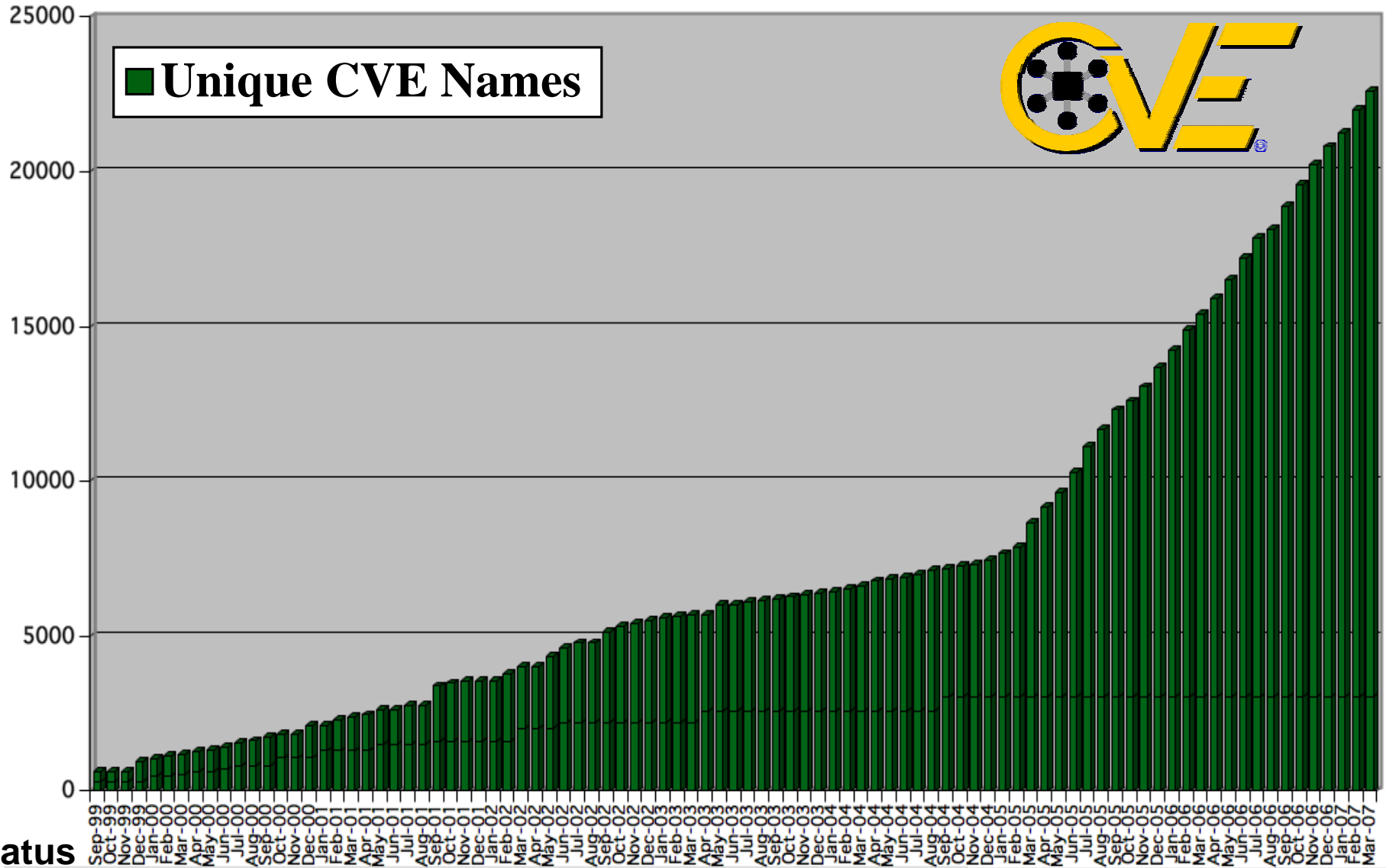
More Information
cwe@mitre.org

[cwe.mitre.org]

Building Consensus About A Common Enumeration



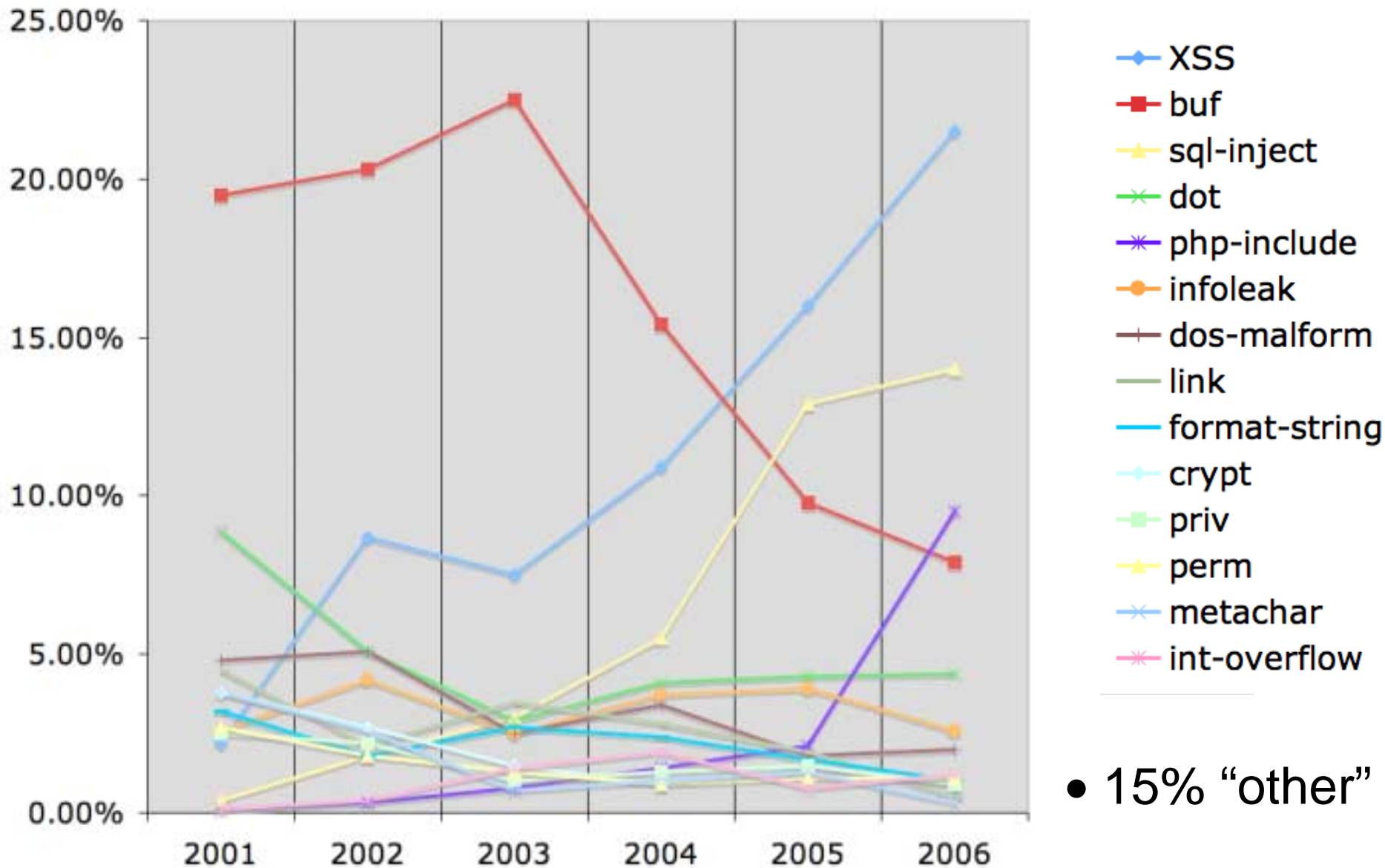
CVE Growth



Status
(as of Feb 28, 2007)

• 22,550 unique CVE names

Vulnerability Type Trends: A Look at the CVE List (2001 - 2006)



Removing and Preventing the Vulnerabilities Requires More Specific Definitions...

—• XSS

—■ buf

—★ sql-inject

—✕ dot

—✱ php-include

—● infoleak

—+ dos-malform

— link

— format-string

— crypt

—■ priv

—★ perm

—✕ metachar

—✱ int-overflow

Cross-site scripting (XSS):

- Basic XSS
- XSS in error pages
- Script in IMG tags
- XSS using Script in Attributes
- XSS using Script Via Encoded URI Schemes
- Doubled character XSS manipulations, e.g. '<<script'
- Invalid Characters in Identifiers
- Alternate XSS syntax

Buffer Errors

- Unbounded Transfer ('classic overflow')
- Write-what-where condition
- Boundary beginning violation ('buffer underwrite')
- Out-of-bounds Read
- Wrap-around error
- Unchecked array indexing
- Length Parameter Inconsistency
- Other length calculation error
- Miscalculated null termination
- String Errors

Relative Path Traversal

- Path Issue - dot dot slash - '../filedir'
- Path Issue - leading dot dot slash - '/../filedir'
- Path Issue - leading directory dot dot slash - '/directory/../filename'
- Path Issue - directory doubled dot dot slash - 'directory/../../filename'
- Path Issue - dot dot backslash - '..\filename'
- Path Issue - leading dot dot backslash - '\..\filename'
- Path Issue - leading directory dot dot backslash - 'directory..\filename'
- Path Issue - directory doubled dot dot backslash - 'directory\..\filename'
- Path Issue - triple dot - '...'
- Path Issue - multiple dot - '....'
- Path Issue - doubled dot dot slash - '...//'
- Path Issue - doubled triple dot slash - '...//'

... which led to the Preliminary List of Vulnerability Examples for Researchers (PLOVER)

- Initial goal: extend vulnerability auditing checklist
- Collected extensive CVE examples
 - Emphasis on 2005 and 2006
 - Reviewed all issues flagged "other"
- 300 weakness types, 1500 real-world CVE examples
- Identified classification difficulties
 - Primary vs. resultant vulns
 - Multi-factor issues
 - Uncategorized examples
 - Tried to separate attacks from vulnerabilities
- Beginning vulnerability theory
 - Properties
 - Manipulations
 - Consequences

- One of the 3 major sources of CWE

PLOVER:

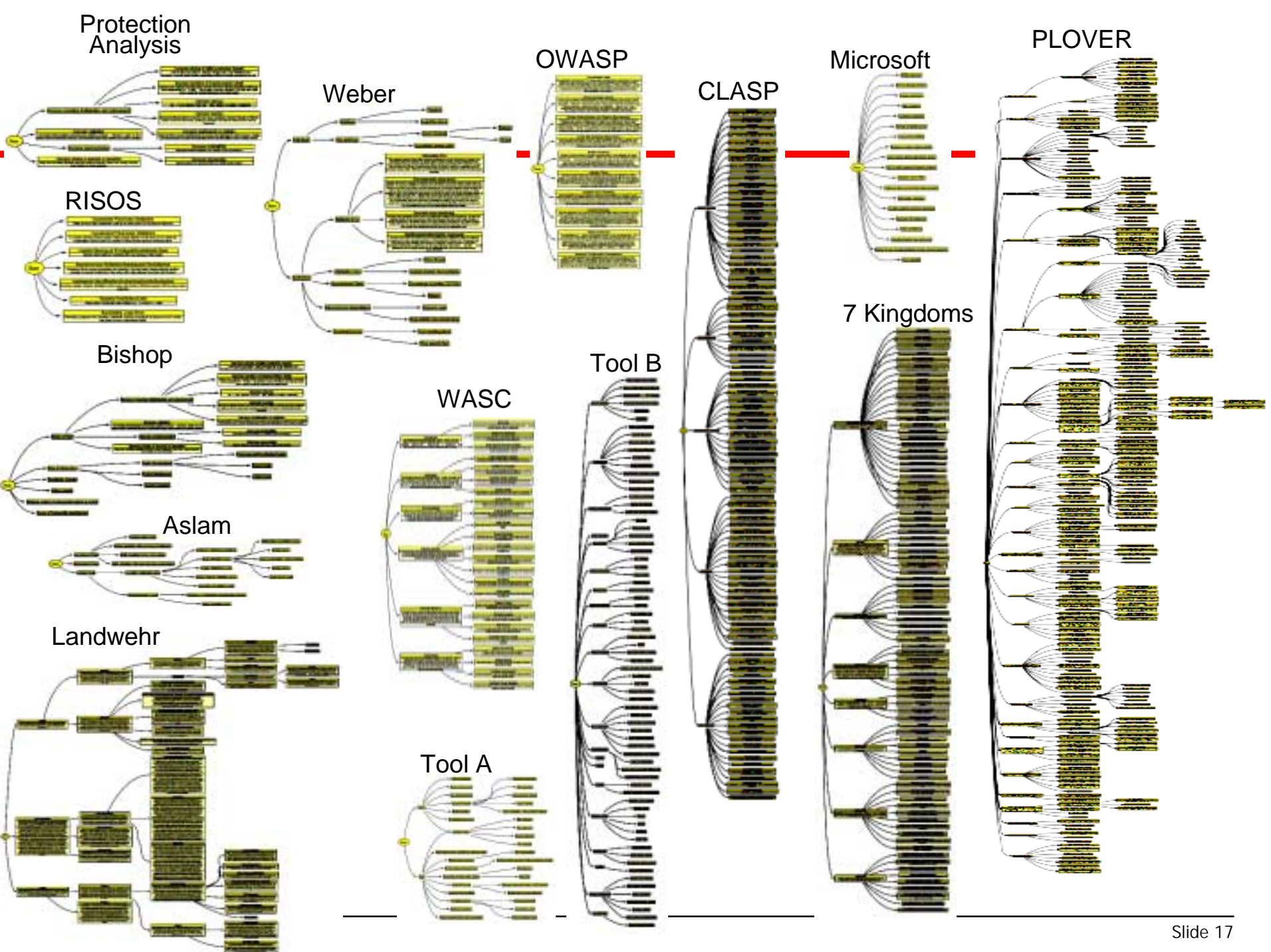
300 “types” of Weaknesses, 1500 real-world CVE examples



| | |
|--|----------|
| [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 |

Where Did We Start?

- Objective: To identify, integrate and effectively describe common software weaknesses known to the industry and software assurance community
- Leveraging taxonomic approach for list integration
 - Identify and review dozens of existing taxonomies
 - Academic and professional (Aslam, RISOS, Landwehr, Bishop, Protection Analysis, etc)
 - High level lists
 - OWASP Top 10, 19 Deadly Sins, WASC, etc.
 - In-depth practical
 - PLOVER, CLASP, 7 Pernicious Kingdoms
 - Create visualizations for effective comparison and analysis
 - Integrating taxonomies
 - Normalizing and deconfliction
 - Finding a proper balance between breadth & depth



Formalizing a Schema for Weaknesses

Identifying Information

- CWE ID
- Name

Describing Information

- Description
- Alternate Terms
- Demonstrative Examples
- Observed Examples
- Context Notes
- Source
- References

Scoping & Delimiting Information

- Functional Area
- Likelihood of Exploit
- Common Consequences
- Enabling Factors for Exploitation
- Common Methods of Exploitation
- Applicable Platforms
- Time of Introduction

Prescribing Information

- Potential Mitigations

Enhancing Information

- Weakness Ordinality
- Causal Nature
- Related Weaknesses
- Taxonomy Mapping
- Research Gaps

CWE-79 Cross-site scripting (XSS)

[cwe.mitre.org/data/definition/79.html]

Individual CWE Dictionary Definition (draft 5)

Cross-site scripting (XSS)

| | |
|------------------------------|--|
| CWE ID | 79 |
| Description | Cross-site scripting weakness occurs when dynamically generated web pages display input, such as login information, that is not properly validated, allowing an attacker to embed malicious scripts into the generated page and then execute the script on the machine of any user that views the site. If successful, Cross-site scripting vulnerability can be used to hijack sessions, create cookies, create requests on behalf of the user, compromise or deface web pages, and execute code on the end user's machine. |
| Alternate Terms | "CSS" was once used but caused significant confusion with the "C" in CSS, significantly, and its use is discouraged. |
| Likelihood of Exploit | High to Very High |
| Weakness Ordinality | Resultant (Weakness Weaknesses) |
| Causal Nature | Explicit (This is an explicit developer error) |
| Common Consequences | Confidentiality: The malicious script involves the disclosure of sensitive information. Access control: In some cases, the script can execute code on a victim's computer, leading to other flaws. |
| Potential Mitigations | Carefully check each input against the specification (white list). All input should be sanitized, but all data should be validated. For headers, the URL itself is not validated, but continuing XSS vulnerabilities exist. |

References

M. Howard and D. LeBlanc. Writing Secure Code. 2nd edition. Microsoft, 2003.

Node Relationships

Child Of - [Injection](#) (74)
Results In - [Mobile Code: Invoking untrusted mobile code](#) (494)
Parent Of - [Basic XSS](#) (80)
Parent Of - [XSS in error pages](#) (81)
Parent Of - [Script in IMG tags](#) (82)
Parent Of - [XSS using Script in Attributes](#) (83)
Parent Of - [XSS using Script Via Encoded URI Schemes](#) (84)
Parent Of - [Doubled character XSS manipulations, e.g. '<<script'](#) (85)
Parent Of - [Invalid Characters in Identifiers](#) (86)
Parent Of - [Alternate XSS syntax](#) (87)
Parent Of - [Mobile Code: Invoking untrusted mobile code](#) (494)

Source Taxonomies

PLOVER - Cross-site scripting (XSS)
7 Pernicious Kingdoms - Cross-site Scripting
CLASP - Cross-site scripting

Applicable Platforms

C
C++
Java
.NET

CWE Cross-Section: 20 of the Usual Suspects

- Absolute Path Traversal (CWE-36)
- Cross-site scripting (XSS) (CWE-79)
- Cross-Site Request Forgery (CSRF) (CWE-352)
- CRLF Injection (CWE-93)
- Error Message Information Leaks (CWE-209)
- Format string vulnerability (CWE-134)
- Hard-Coded Password (CWE-259)
- Insecure Default Permissions (CWE-276)
- Integer overflow (wrap or wraparound) (CWE-190)
- OS Command Injection (shell metacharacters) (CWE-78)
- PHP File Inclusion (CWE-98)
- Plaintext password Storage (CWE-256)
- Race condition (CWE-362)
- Relative Path Traversal (CWE-23)
- SQL injection (CWE-89)
- Unbounded Transfer ('classic buffer overflow') (CWE-120)
- UNIX symbolic link (symlink) following (CWE-61)
- Untrusted Search Path (CWE-426)
- Weak Encryption (CWE-326)
- Web Parameter Tampering (CWE-472)

CWE Cross-Section: 22 More Suspects

- **Design-Related**

- High Algorithmic Complexity (CWE-407)
- Origin Validation Error (CWE-346)
- Small Space of Random Values (CWE-334)
- Timing Discrepancy Information Leak (CWE-208)
- Unprotected Windows Messaging Channel ('Shatter') (CWE-422)
- Inherently Dangerous Functions, e.g. gets (CWE-242)
- Logic/Time Bomb (CWE-511)

- **Low-level coding**

- Assigning instead of comparing (CWE-481)
- Double Free (CWE-415)
- Null Dereference (CWE-476)
- Unchecked array indexing (CWE-129)
- Unchecked Return Value (CWE-252)
- Path Equivalence - trailing dot - 'file.txt.' (CWE-42)

- **Newer languages/frameworks**

- Deserialization of untrusted data (CWE-502)
- Information leak through class cloning (CWE-498)
- .NET Misconfiguration: Impersonation (CWE-520)
- Passing mutable objects to an untrusted method (CWE-375)

- **Security feature failures**

- Failure to check for certificate revocation (CWE-299)
- Improperly Implemented Security Check for Standard (CWE-358)
- Failure to check whether privileges were dropped successfully (CWE-273)
- Incomplete Blacklist (CWE-184)
- Use of hard-coded cryptographic key (CWE-321)

... and about
550 more

Where Are We Today?

Quality

- “Kitchen Sink” – In a good way
 - Many taxonomies, products, perspectives
 - Varying levels of abstraction
 - Directory traversal, XSS variants
- Mixes attack, behavior, feature, and flaw
 - Predominant in current research vocabulary, especially web application security
 - Complex behaviors don’t have simple terms
 - New/rare weaknesses don’t have terms

Quantity

- Draft 5 - over 600 entries
- Currently integrating content from top 15 – 20 tool vendors and security weaknesses “knowledge holders” under NDA

Accessibility

- Website is live with:
 - Historical materials, papers, alphabetical full enumeration, taxonomy HTML tree, CWE in XML, ability to URL reference individual CWEs, etc

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.



Current Community Contributing to the Common Weakness Enumeration

- AppSIC
- Cenxic
- CERIAS/Purdue University
- CERT/CC
- Cigital
- CodescanLabs
- Core Security
- Coverity
- DHS
- Fortify
- IBM Interoperability Clearing House
- JHU/APL
- JMU
- Kestrel Technology
- KDM Analytics
- Klocwork
- McAfee/Foundstone
- Microsoft
- MIT Lincoln Labs
- MITRE
- North Carolina State University
- NIST
- NSA
- Oracle
- Ounce Labs
- OWASP
- Palamida
- Parasoft
- PolySpace Technologies
- proServices Corporation
- SecurityInnovation
- Secure Software
- Security University
- Semantic Designs
- SofCheck
- SPI Dynamics
- SureLogic, Inc.
- UNISYS
- VERACODE
- Watchfire
- WASC
- Whitehat Security, Inc.
- Tim Newsham

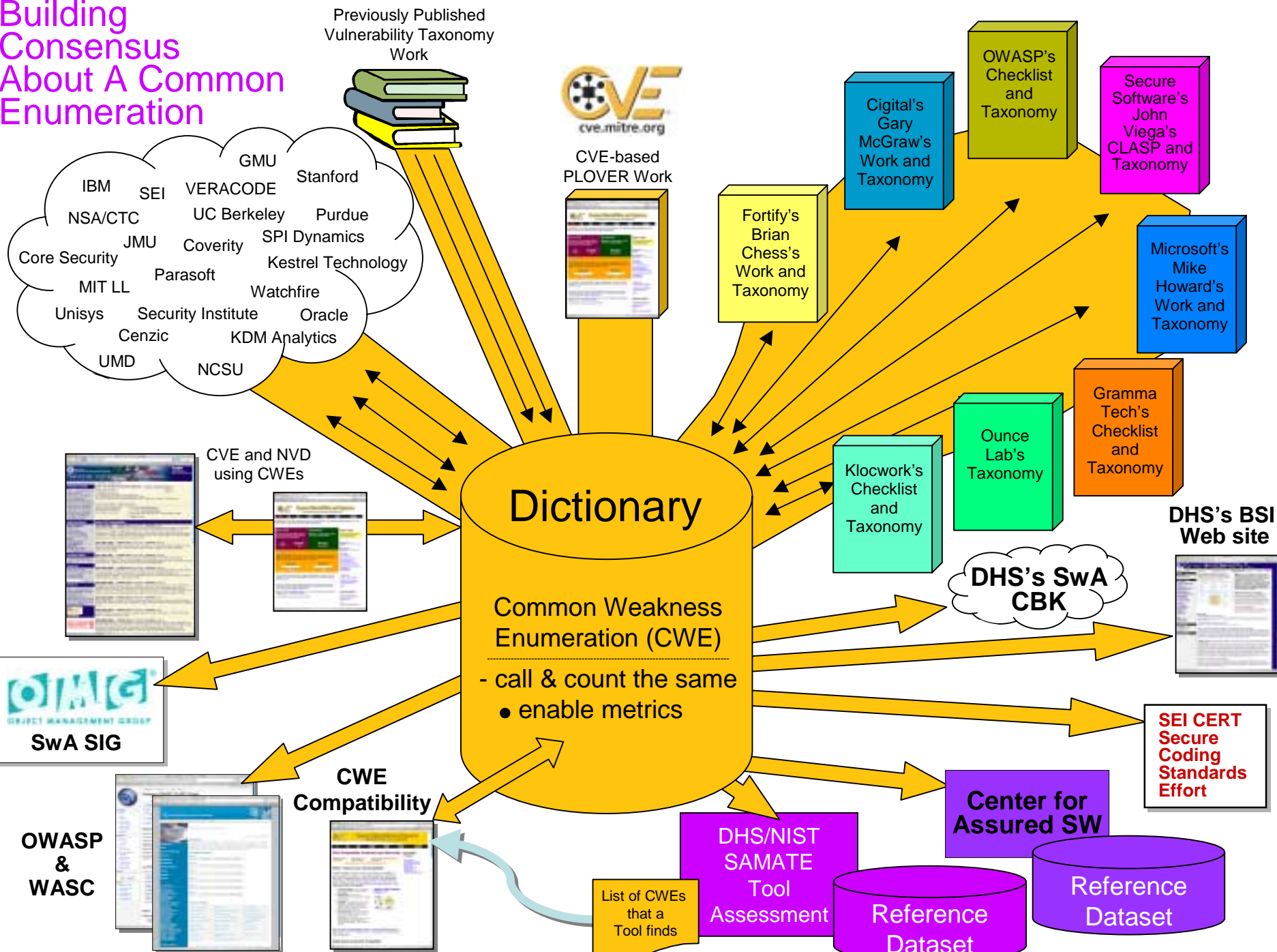
Planned Improvements - Content

- Metadata tagging
 - Language, OS, etc.
 - Time of Introduction
 - Vulnerability theory
 - Other ideas?
- Content cleanup
 - Consistent naming
 - Structural refactoring
 - Attack-centric wording (align to CAPEC)
- Formalization
 - SBVR

Planned Improvements - Site Usability

- Search
 - Select a subset of the catalog using any of the metadata
 - Display results and make available as XML
 - Predefined searches
- Graphical Visualization
 - Dynamic adjustment and navigation
 - Alternate taxonomies

Building Consensus About A Common Enumeration



CWE-Compatible & CWE-Effective

CWE Compatible:

1. CWE-compatible “intent” declared
 - vendor with shipping product declares intent to add support for CWE ids
2. CWE-compatible “output and searchable” declared
 - vendor declares that their shipping product provides CWE ids and supports searching
3. CWE-compatible “mapping accuracy” compatibility questionnaire posted
 - questionnaire for mapping accuracy posted to CWE web site
4. CWE-compatible means it meets the following requirements:
 - Can find items by CWE id (CWE searchable)
 - Includes CWE id in output for each item (CWE output)
 - Explain the CWE functionality in their item’s documentation (CWE documentation)
 - Provided MITRE with “weakness” item mappings to validate the accuracy of the product or services CWE ids
 - Makes a good faith effort to keep mappings accurate

CWE-Effective:

1. CWE-effectiveness list posted
 - CWE ids that the tool is declaring “effectiveness for” is posted to CWE web site
2. CWE-effectiveness test results posted
 - CWE test cases obtained from NIST reference data set generator by tool owner
 - Scoring sheet for requested CWE test cases provided to MITRE by NIST
 - Tool results from evaluating CWE-based sample applications (CWE test cases) provided to MITRE for processing and posting

CWE Compatibility and Effectiveness Program Launched

CWE Common Weakness Enumeration
A community-developed dictionary of common software weaknesses

Name > Compatibility View the CWE List

CWE List
Full Dictionary View
Classification Tree
Other Views

About
Sources
Process
Documents

Community
Related Activities

News
Calendar

Compatibility
Program
Requirements
Make a Declaration

Contact Us
Search the Site

CWE Compatibility

The CWE Compatibility and Effectiveness Program provides for a product or service to be reviewed and registered as officially "CWE-Compatible" and "CWE-Effective," thereby assisting organizations in their selection and evaluation of tools and/or services for assessing their acquired software for known types of weaknesses and flaws, for learning about the various weaknesses and their possible impact, or to obtain training and education about these issues. Organizations with products and services still working towards compatibility and effectiveness are also listed.

CWE-Compatible Products and Services must meet the first four (4) of the six (6) requirements below, while CWE-Effective Products and Services must meet all six (6) requirements. Please review the [complete set of requirements](#) to fully understand CWE compatibility and effectiveness.

- CWE Searchable** — users may search security elements using CWE identifiers
- CWE Output** — security elements presented to users includes, or allows users to obtain, associated CWE identifiers
- Mapping Accuracy** — security elements accurately link to the appropriate CWE identifiers
- CWE Documentation** — capability's documentation describes CWE, CWE compatibility, and how CWE-related functionality in the capability is used
- CWE Coverage** — for CWE-Effectiveness, capability's documentation explicitly lists the CWE identifiers that the capability is effective at locating in software
- CWE Test Results** — for CWE-Effectiveness, test results from the capability showing the results of assessing software for the CWEs are posted on the CWE Web site

See the [CWE Compatibility and Effectiveness Program](#) or email cwe@mitre.org for information on how to register your product(s) or services(s) as CWE-compatible effective.

Section Contents
Compatibility
Program
Requirements
Make a Declaration

cwe.mitre.org/compatible/

Homeland Security

CWE is sponsored by the [U.S. Department of Homeland Security](#).
This Web site is hosted by [The MITRE Corporation](#).
Copyright 2007, The MITRE Corporation. CWE and the CWE logo are trademarks of The MITRE Corporation.
Contact: ped@mitre.org for more information.

Page Last Updated: December 29, 2006
[Privacy Policy](#)
[Terms of Use](#)
[Contact Us](#)

CWE Compatibility and Effectiveness Process Posted

The screenshot shows a web browser window with the URL <http://cwe.mitre.org/compatible/program.html>. The page title is "CWE - CWE Compatibility and Effectiveness Program". The main heading is "CWE Compatibility and Effectiveness Program" with the subtitle "A community-developed dictionary of common software weaknesses". The page is divided into several sections: "Introduction", "Phase 1 - Declaration Phase", and "Phase 2 - Evaluation Phase". A navigation menu on the left includes "CWE List", "About", "Community", "News", "Calendar", "Compatibility", and "Contact Us". A "Section Contents" menu on the right lists "Compatibility", "Program", "Requirements", and "Make a Declaration".

CWE Compatibility and Effectiveness Program

Introduction

The CWE Compatibility and Effectiveness Program is a formal review and evaluation process for organizations wishing to declare their information security products and services as CWE-Compatible and CWE-Effective and have them formally evaluated.

Compatible and Effective products and services, as well as those working towards compatibility and effectiveness, will be posted on the "CWE-Compatible and Effective Products and Services" page on the CWE Web site and included on handouts at information security and related tradeshows and events at which MITRE exhibits CWE (see the [CWE Calendar](#)).

The formal CWE Compatibility and Effectiveness Program includes three phases: Declaration, Evaluation, and Effectiveness.

Phase 1 - Declaration Phase

The Declaration Phase requires the completion of a short informational "CWE Compatibility Declaration Form" used to register an organization's declaration of intent with respect to CWE compatibility and effectiveness. In this phase you are asked to review the compatibility and effectiveness requirements and then make a statement regarding whether your organization believes that its product or service currently fulfills the compatibility requirements, or if your organization is working towards fulfilling the requirements. This phase of the CWE compatibility and effectiveness process does not result in an official evaluation or assessment by MITRE; rather, MITRE only reviews the declaration. As long as the products or services are commercially or publicly available, the declaration and an endorsement quote from you (if desired) is posted on the CWE Web site.

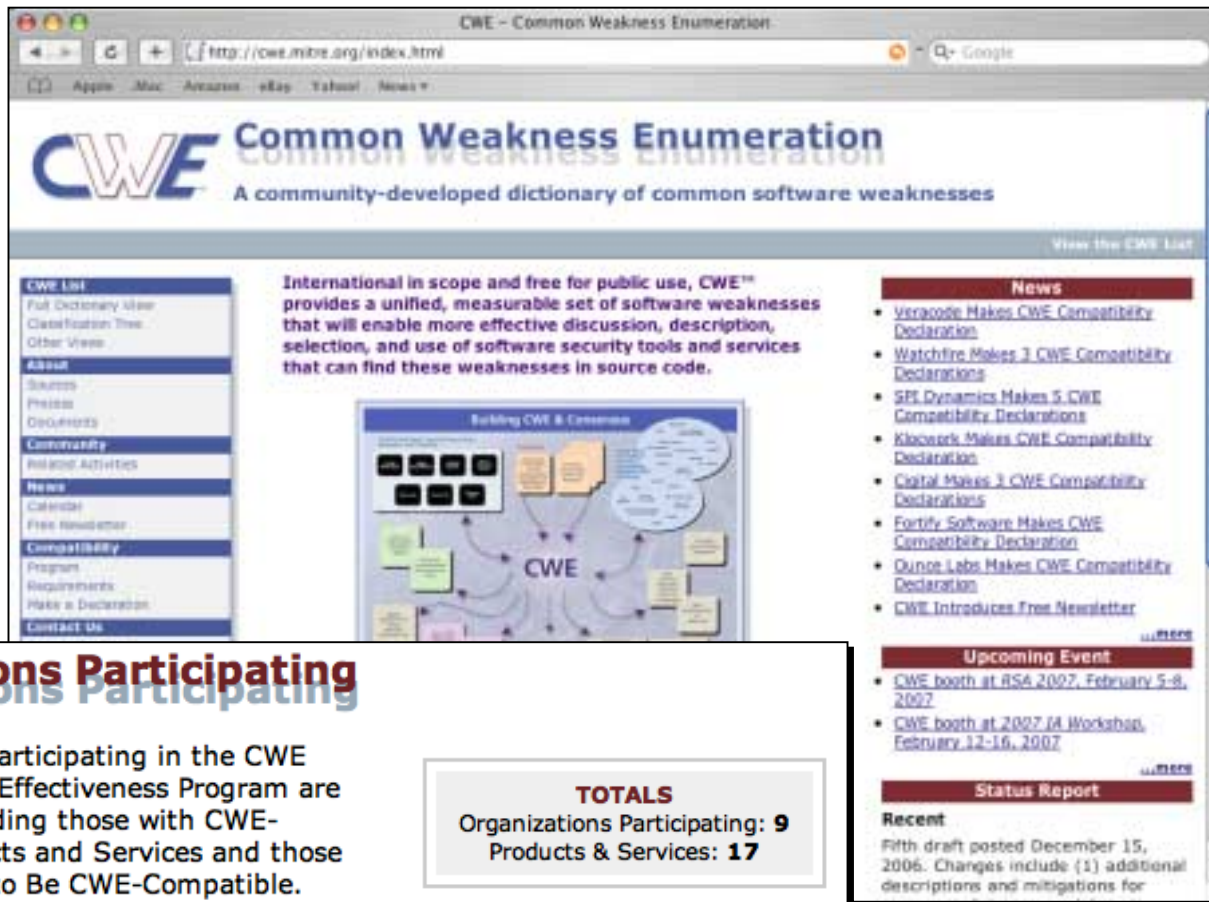
Phase 2 - Evaluation Phase

The Evaluation Phase requires completion of Phase 1 output, CWE searchable, and CWE documentation. "Compatibility Requirements Evaluation Form" that includes several evaluation activities. You will receive an "Organization Welcome Kit" with items for your Web site.

This formal evaluation process includes a "branding program" and logo to indicate successful completion of the compatibility and effectiveness evaluation. A major component of this phase requires specific details about how your organization has satisfied each of the mandatory requirements in the [Requirements and Recommendations for CWE Compatibility and CWE Effectiveness](#) document. The Phase 2 "CWE Compatibility Requirements Evaluation Form" also requires the signature

cwe.mitre.org/compatible/program.html

CWE Compatibility and Effectiveness Declarations Posted



The screenshot shows the CWE website with the following content:

- CWE List**: Full Dictionary view, Classification Tree, Other Views
- About**: Sources, Press, Documents
- Compatibility**: Related Activities
- News**: Calendar, Free Newsletter
- Compatibility**: Program Requirements, Make a Declaration, Contact Us

International in scope and free for public use, CWE™ provides a unified, measurable set of software weaknesses that will enable more effective discussion, description, selection, and use of software security tools and services that can find these weaknesses in source code.

Building CWE & Cessmate

News

- Veracode Makes CWE Compatibility Declaration
- Watchfire Makes 3 CWE Compatibility Declarations
- SPI Dynamics Makes 5 CWE Compatibility Declarations
- Kloosock Makes CWE Compatibility Declaration
- Castal Makes 3 CWE Compatibility Declarations
- Fortify Software Makes CWE Compatibility Declaration
- Dunco Labs Makes CWE Compatibility Declaration
- CWE Introduces Free Newsletter

Upcoming Event

- CWE booth at RSA 2007, February 5-8, 2007
- CWE booth at 2007 (A) Workshop, February 12-16, 2007

Status Report

Recent

Fifth draft posted December 15, 2006. Changes include (1) additional descriptions and mitigations for

Organizations Participating

All organizations participating in the CWE Compatibility and Effectiveness Program are listed below, including those with CWE-Compatible Products and Services and those with Declarations to Be CWE-Compatible.

TOTALS

Organizations Participating: **9**
Products & Services: **17**

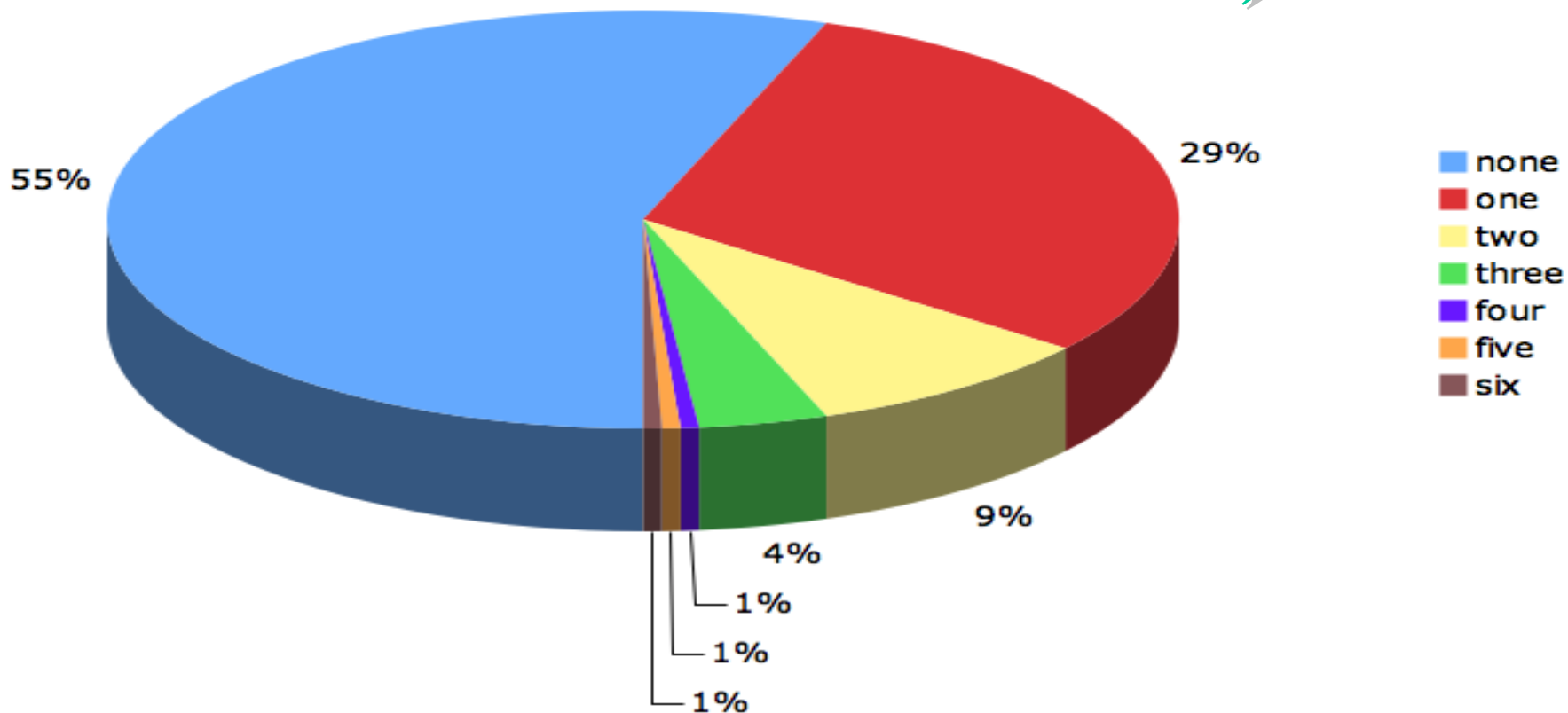
Products are listed alphabetically by organization name:

cwe.mitre.org/compatible/organizations.html

Coverage of CWE

CWE

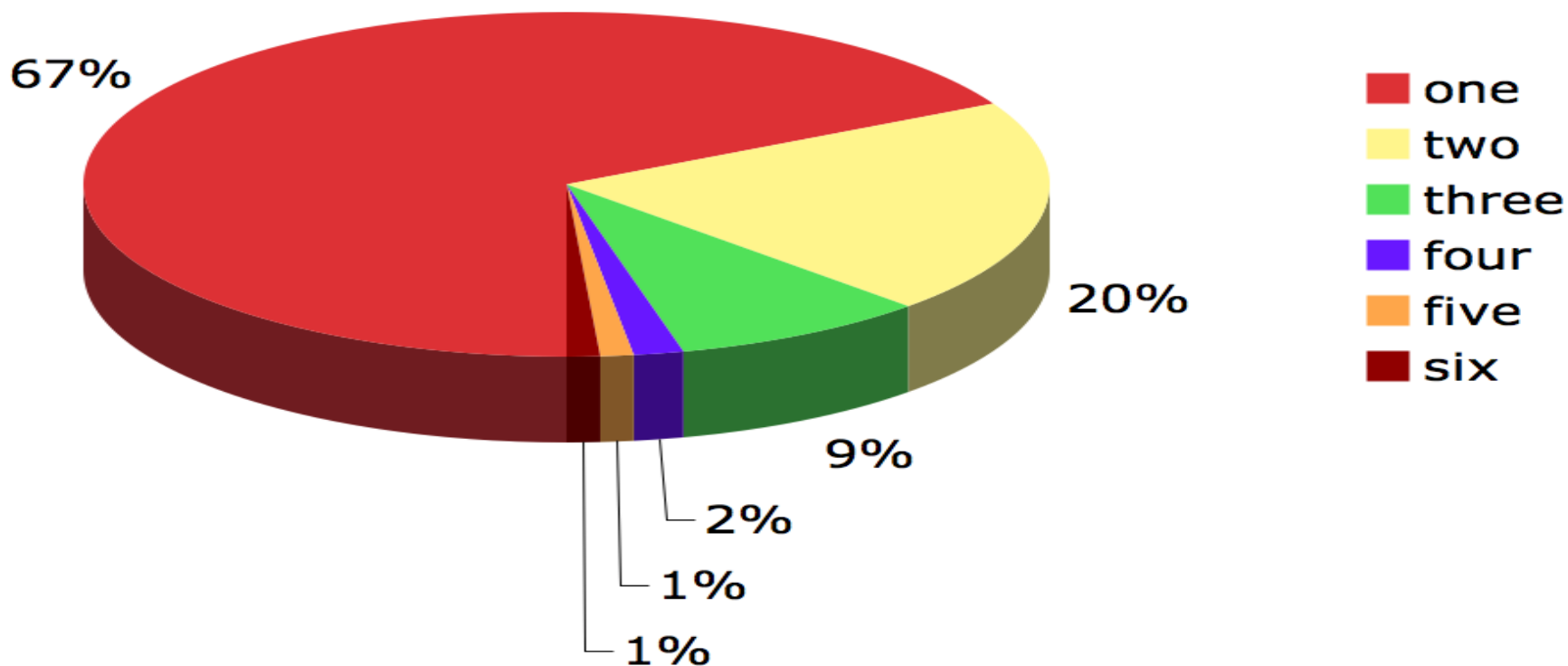
Draft



Covered CWEs - By Number of Tools

Covered CWEs

Draft

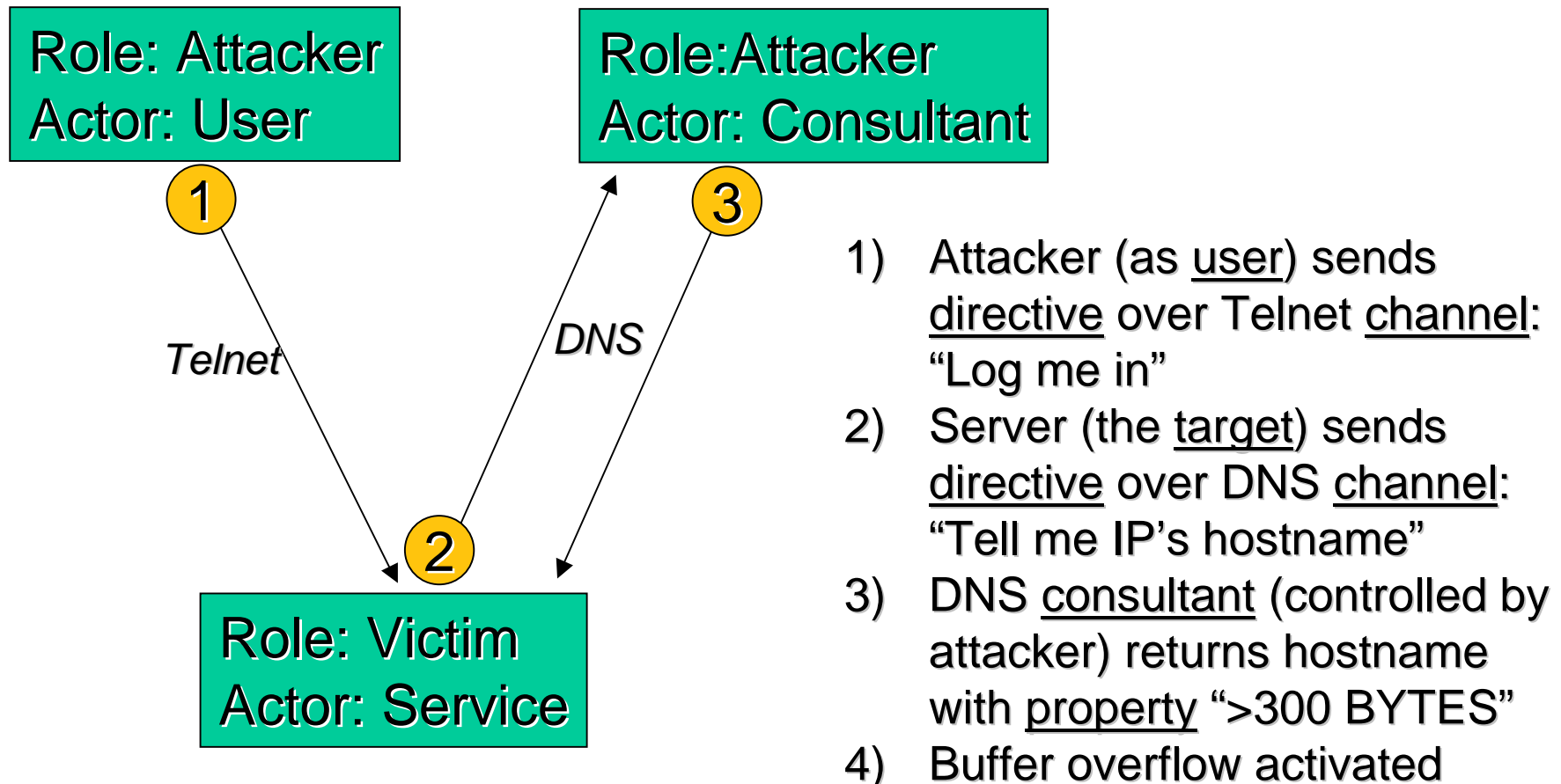


The Path to Formalization -- Vulnerability Theory: Problem Statement and Rationale

- With 600+ variants, what are the main themes?
- Why is it so hard to classify vulnerabilities cleanly?
 - CWE, Pernicious Kingdoms, OWASP, others have had similar difficulties
- Same terminology used in multiple dimensions
 - Frequent mix of attacks, threats, weaknesses/faults, consequences
 - E.g. buffer overflows, directory traversal
- Goal: Increase understanding of vulnerabilities
 - Vocabulary for more precise discussion
 - Label current inconsistencies in terminology and taxonomy
 - Codify some of the researchers' instinct
- One possible application: gap analysis, defense, and design recommendations
 - “Algorithms X and Y both assume input has property P. Attack pattern A manipulates P to compromise X. Would A succeed against Y?”
 - “Technology Z has properties P1 and P2. What vulnerability classes are most likely to be present?”
 - “Why is XSS so obvious but so hard to eradicate?”

Some Basic Concepts of Vulnerability Theory: By Example

Buffer overflow using long DNS response



Artifact Labels

- Artifact: an observable segment of code, design, or algorithm
- Interaction Point (“Entry point”)
 - A relevant point within the code/design where a user interacts with the code/design
 - Associated with a channel
 - Why not “entry point?” Overlaps reverse engineering terms.
- Intermediate Fault
 - A behavior by the code/design that influences future behavior
 - Root cause?
- Crossover point
 - The first point where expected properties are violated
 - Sometimes IN BETWEEN lines of code (missing protection scheme)
- Control Transfer Point
 - The first point beyond which the program cannot prevent a security violation
- Activation Point
 - The point where the “payload” is activated and performs the actions intended by the attacker
- Resultant Fault
 - A fault after a “Primary” fault that is also where incorrect behavior occurs; could be an activation point

Artifact Labels - Example

```
1  print HTTPResponseHeader;
2  print "<title>Hello World</title>";
3  ftype = HTTP_Query_Param("type");
-----
4  str = "/www/data/";
-----
5  strcat(str, ftype); strcat(str, ".dat");
-----
6  handle = fileOpen(str, "read");
-----
7  while((line=readFile(handle)))
8  {
9    line=stripTags(line, "script");
-----
10  print line;
-----
11  print "<br>\n";
12  }
13  close(handle);
```

OVAL/XCCDF/
CCE/CPE

System Assurance
Guidance/
Mandates/
Requirements

CWE/CAPEC/
SBVR

Certification &
Assessment of System
Development,
Integration,
&
Sustainment
Activities

Software Assurance

Mission:
Our Mission Statement

Contact:
Co-Chairs: Ms. Diana KIM Analyst, Ms. J.E. Ben MA Systems

The primary mission of the SWA SIG is to work with Platform and Domain Task Forces and other software industry entities and groups external to the DRAG, to coordinate the establishment of a common framework for analysis and exchange of information related to software assurance. To that end, the goal of the SWA SIG is to facilitate the development of a transformation for a Software Assurance Framework that will:

- Establish a common framework of software products that can be used to represent any/all classes of software to software suppliers and consumers to represent their claims and arguments/respectively along with the corresponding evidence, employ automated tools (to address scale)
- Verify that products have sufficiently satisfied characteristics in advance of product acquisition that system engineers/programs can use the products to build (compose) larger assumed systems with them
- Create industry to improve visibility into the status of software assurance during development

NIST Draft Special Publication 500-268

**Source Code Security Analysis Tool
Functional Specification Version 1.0**

Information Technology Laboratory (ITL), Software Diagnostics and Conformance Testing Division

SAMATE Reference Dataset

Software Assurance Method and Tool Evaluation

Welcome to the NIST SAMATE Reference Dataset Project

CS 390S: Secure Programming

Slides will be posted on the same day as the class.

February 21: Over calls, That Week 7 (pdf) View 7 presentations

CWE Coverage starts local issues covered.

PURDUE UNIVERSITY

CS390S, Week 7: Calling Programs, Trust

Patrice Maurier, Ph.D., M.Sc., CISSP
February 21, 2007

Developed thanks to support and contributions from Symantec Corporation, support from the NSF SP5 Capacity Building Program (Award Number 0113725) and the Purdue e-Enterprise Center
Copyright ©2007 Purdue Research Foundation. All rights reserved.

CWE - Organizations Participating

Common Weakness Enumeration
A community-developed dictionary of common software weaknesses

Organizations Participating

All organizations participating in the CWE Compatibility and Effectiveness Program are listed below, including those with CWE-Compatible Products and Services and those with Declarations to Be CWE-Compatible.

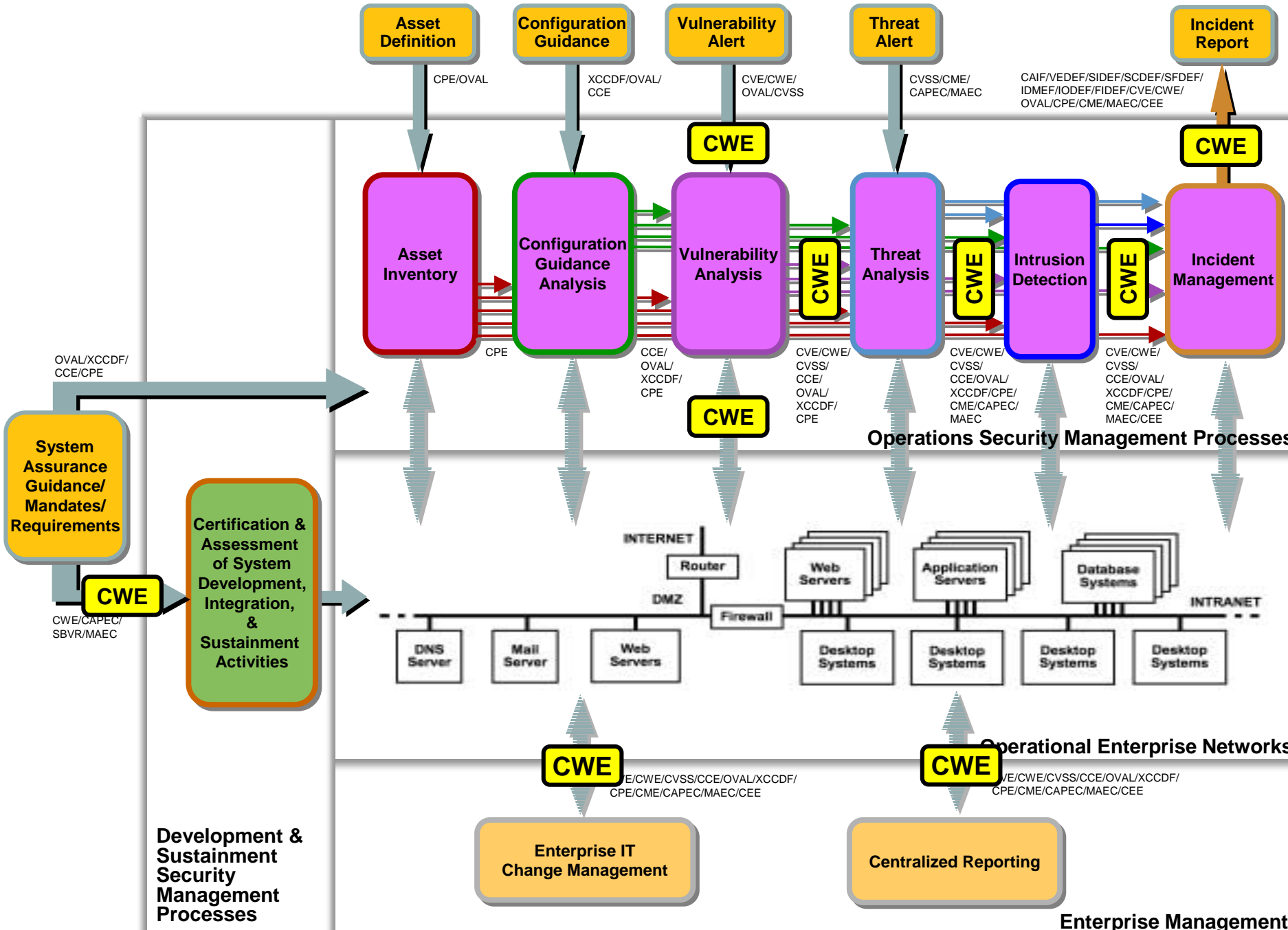
Products are listed alphabetically by organization name:

TOTALS
Organizations Participating: **7**
Products & Services: **15**

A B C D E F G H I J K L M N O P

...ance task developers with a
...ethods. These test cases are
...sion) (revelation, "synthesis"
...al software applications with
...ts, languages, platforms, and
...y. We hope more information
...ase repository, or download

Knowledge Repositories



The Road Ahead for the CWE effort

- Finish the strawman dictionary/taxonomy
- Create a web presence
- Get NDAs with knowledgeable organizations
- Merge information from NDA'd sources
- Get agreement on the detailed enumeration
- Dovetail with test cases (NIST/CAS)
- Dovetail with attack patterns (Cigital)
- Dovetail with coding standards (SEI CERT/CC)
- Dovetail with BSI, CBK, OMG SwA SIG, ISO/IEC,...
- Create alternate views into the CWE dictionary
- Establish CWE Editorial Board (roles & members)
- Establish CWE Compatibility Requirements
- Collect CWE Compatible Declarations
- Vulnerability Theory --> Formalization

DONE
DONE
DONE
In Process
In Process
In Process
In Process
In Process
In Process
In Process
Pending
In Process
Drafted
Started
Started