Being Explicit About Weaknesses

Robert A. Martin - MITRE
Sean Barnum - Cigital
Steve Christey - MITRE

1 March 2007

cigital

MITRE

Software Security Assurance

SECURE

Don't Let Your Applications Get You Down

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 th
antivirus software and intrusion de
tems, although essential to their ov
ty posture, no longer provide adeq
tion. Black-hat hackers and othe
intruders are increasingly circumventing networ
curity and exploiting loopholes in applications.

These vulnerabilities are costing enterprises a l
National Institute of Standards & Technology r
companies are spending about $60 billion a year
and correcting software errors. Exploited softwar
the U.S. financial services industry more than $
year, according to NIST.

Part of the problem is the changing nature of ent
ware, says Gartner analyst John Pescatore. Databas
once kept in centralized data centers and accesse
or exclusively by apps within the enterprise's sec
are now connected to customers, partners and supp
the world through the Web.

Another part of the problem is the nature of s
velopment, he says. Application development or
are rewarded for delivering features and meeting
so developers tend to focus on those things. Conse
plication security becomes an afterthought, and de
dress vulnerabilities only if they are discovered a
plications have been developed.

Not doing the security work up front can quic
expensive. Gartner figures it's 50 times more costl

PHOTO-ILLUSTRATION BY ANN ELLIOTT CUTTING

www.adtmag.com

GCN News 10.10.05

Procurement is key to security, IT execs say

BY PATIENCE WAIT | GCN 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 strengthen the country's cyber-infrastructure in the process.

INFORMATION ASSURANCE

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 Purdy, acting director of DHS' National Cyber Security Division.

Vendors will not invest in improving the quality of their software of their own volition, said Priscilla Guthrie, deputy CIO and deputy assistant secretary of Defense for networks and information integration. "We've got to use acquisition organiza-

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

— DOD'S PRISCILLA GUTHRIE

NEWS&ANALYSIS

Update gives developers head start in fixing code

By Paul F. Roberts

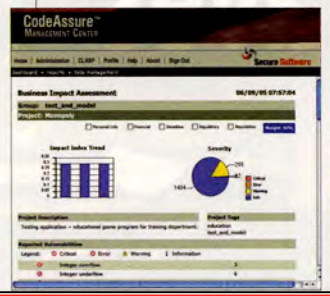
A NEW SOFTWARE RELEASE from Secure Software Inc. is designed to make it easier for organizations doing software development to spot and resolve security flaws in raw computer code.

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 DevInspect and SecureObjects into Visual Studio 2005 and Visual Studio 2005



SC MAGAZINE
FOR IT SECURITY PROFESSIONALS

- Home
- News
- Products
- Features
- White Papers
- Events
- Advertising
- Working with SC

SC MAGAZINE AWARDS 2006

Nominations are now open,click here t

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

DoD Software Assurance Initiative

OSD

Mitchell Komaroff, OASD (NII)/DCIO
Kristen Baldwin, OUSD(AT&L)/DS

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 a 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, 5000.2, 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

NSA

- Offensive side
  - Pedigree problem
  - Interconnectivity
  - Value of IT information
  - Many adversaries
- Defensive side
  - Fragmented efforts
  - Immature science
  - Lack of resources
  - Hard, comp…

…likely to b…
…vernment i…
…s growing…

Software Assurance:

DHS

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

Considerations for Advancing a National Strategy to Secure Cyberspace

Sept 7, 2005

Joe Jarzombek, PMP
Director for Software Assurance
National Cyber Security Division
US Department of Homeland Security

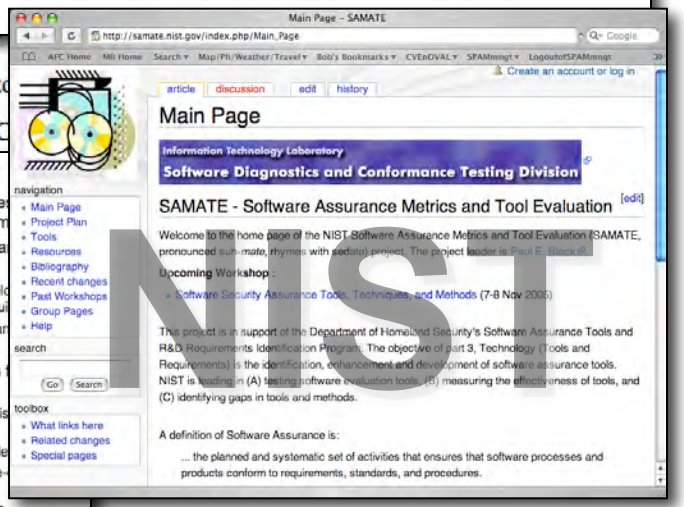
Homeland Security

Driving Needs for Software Assurance

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

Homeland Security

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 National Institute of Standards and Technology • Technology Admin

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.

NIST National Institute of Standards and Technology • Technology Administration • U.S. Department of Commerce

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


SSATTM – SAMATE
http://samate.nist.gov/index.php/SSATTM
Google
AFC Home
MII Home
Search
Map/Ph/Weather/Travel
Bob's Bookmarks
CVEnOVAL
SPAMmngt
Create an account or log in
article
discussion
edit
history
SSATTM
Information Technology Laboratory
Software Diagnostics and Conformance Testing Division
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
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
The Case for Common Flaw Enumeration – Robert Martin, Steven Christey, Joe Jarzombek
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 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)

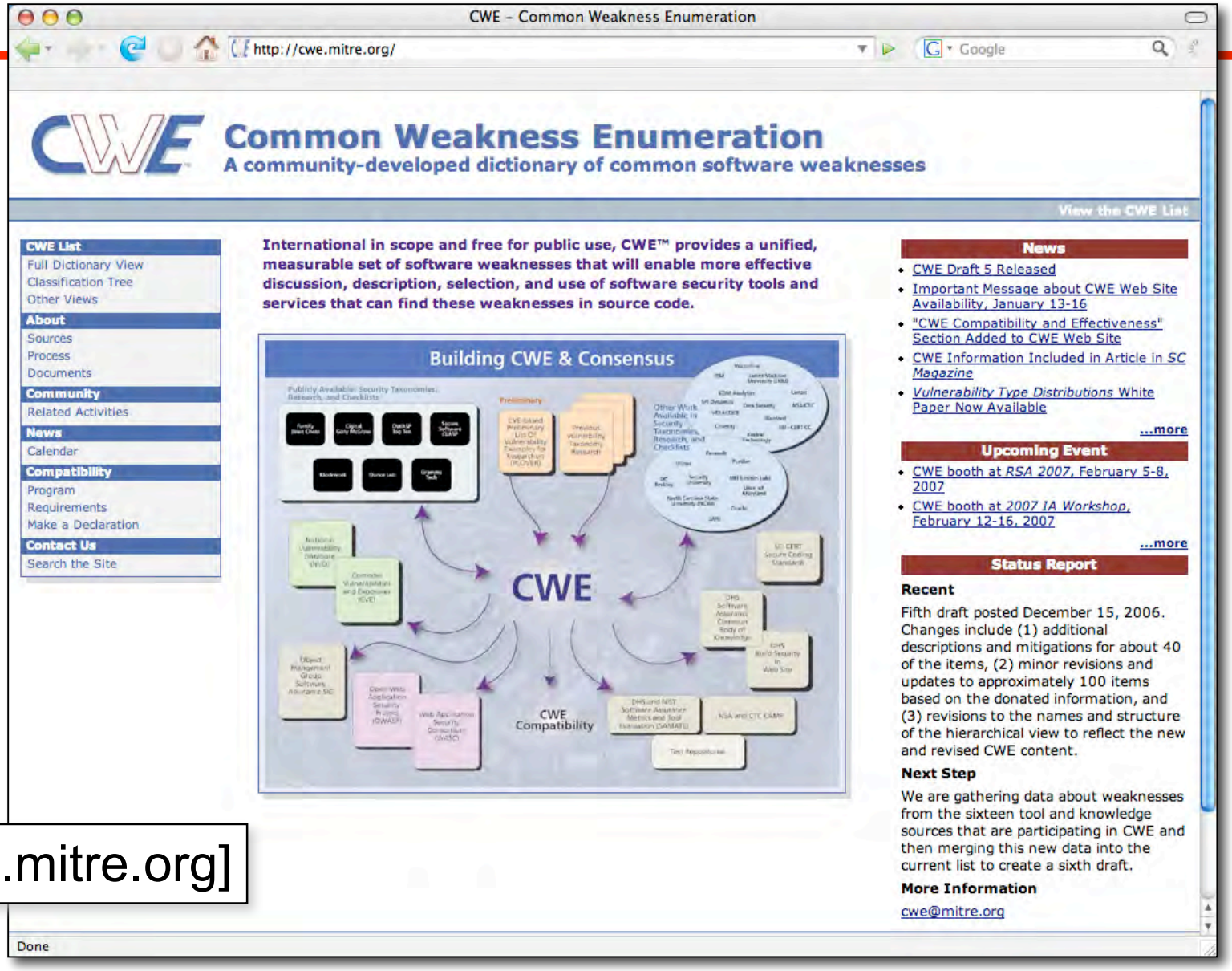
- Systems Development Manager Issue Areas:
  - What are the software weaknesses I need to protect against
    - Architecture, design, code
  - Can I look through the issues by technologies, risks, severity
  - What have the pieces of my system been vetted for?
    - COTS packages, organic development, 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)

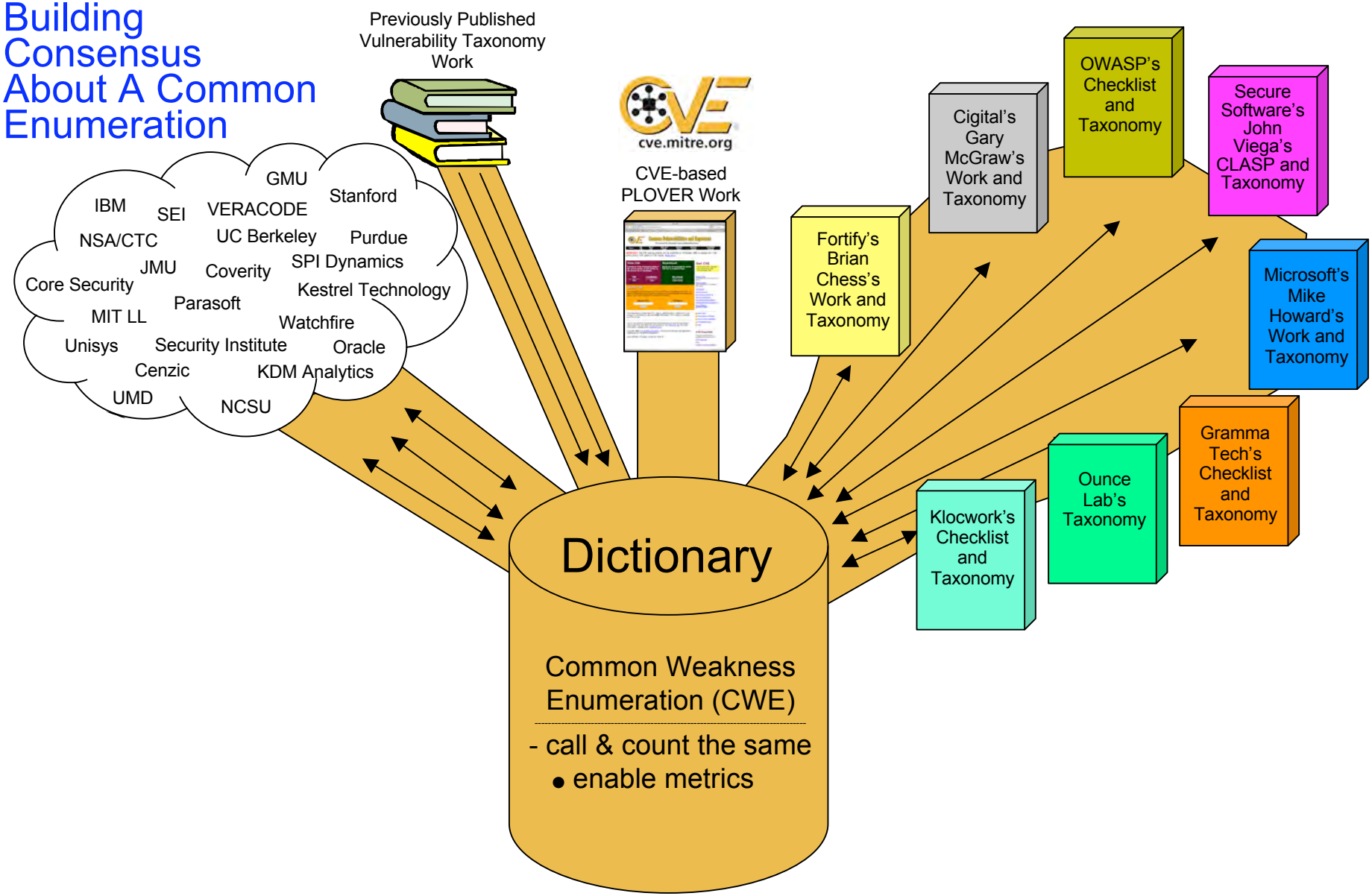
- COTS Product Vendor Issue Areas:
  - What have I vetted my applications for?
  - What do my customers want me to vet for?
- Researcher Issue Areas:
  - Quickly understand what is known
  - Easily identify areas to contribute/refine/correct
- Educator Issue Areas:
  - Train students with the same concepts 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

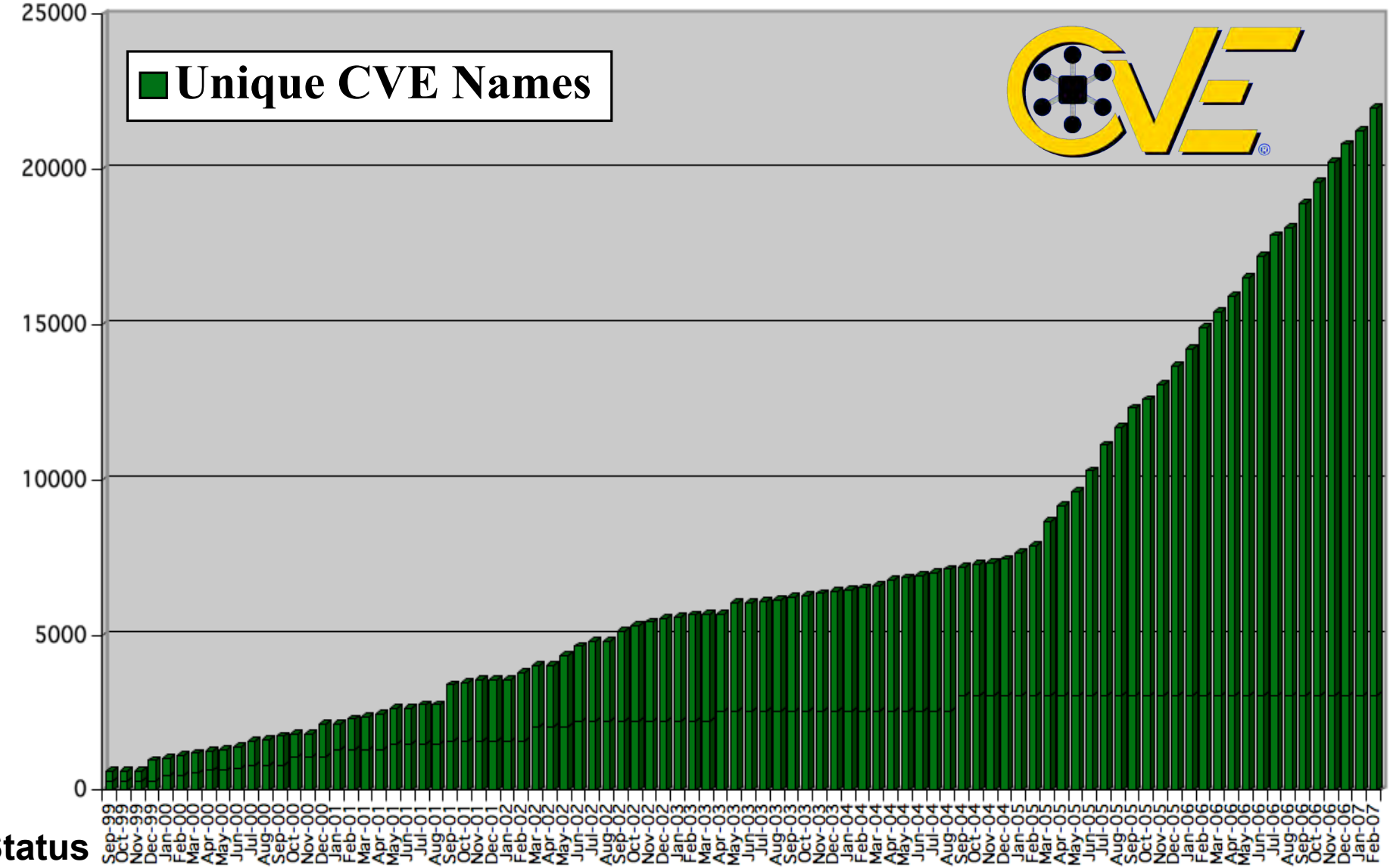


[cwe.mitre.org]

Building Consensus About A Common Enumeration



CVE Growth

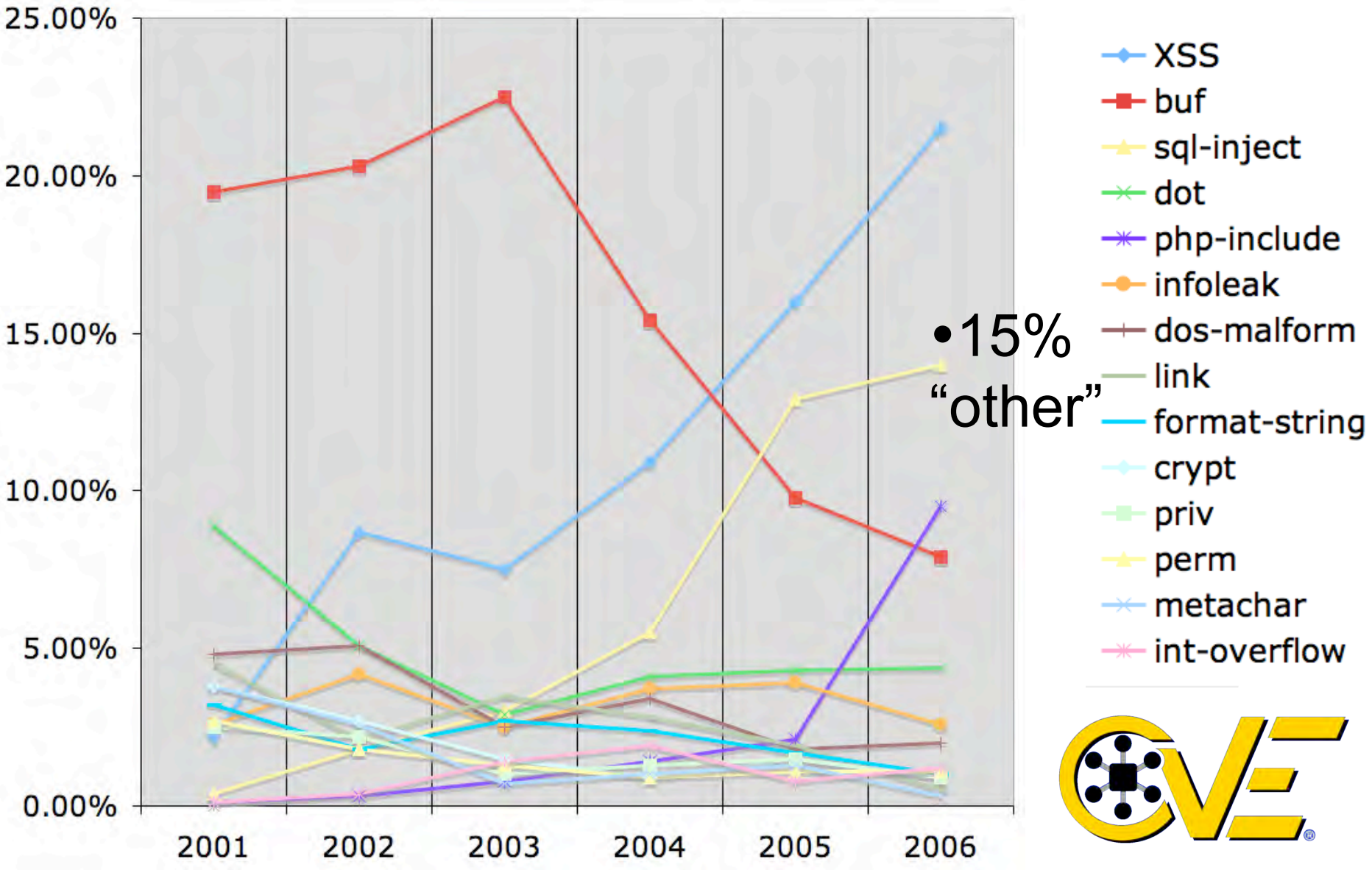


Status

(as of Feb 1, 2007)

- **21,990 unique CVE names**

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

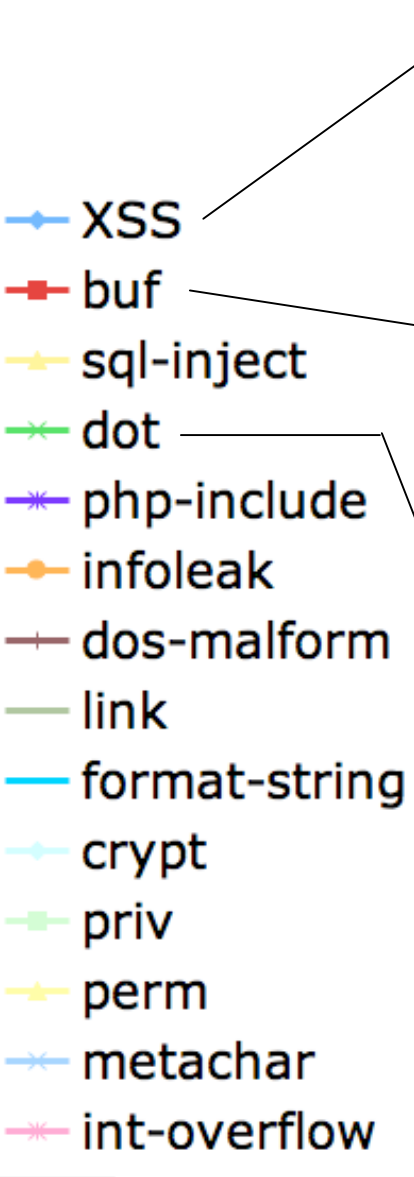


- 15% "other"

But…

- What about the 15% “Other” in 2006?
  - What is up-and-coming? What’s important but below the radar?
- Variants matter in evaluating software quality
  - Example: obvious XSS vs. non-standard browser behaviors that bypass filters
- Bug X might be “resultant from” or “primary to” Bug Y, yet both are thought of as vulnerabilities
  - E.g. integer overflows leading to buffer overflows
  - How can we tell if things are improving?
- Maybe some issues are symptoms of deeper problems
  - *Error: Couldn’t open file “lang-<SCRIPT>alert(‘XSS’)</SCRIPT>.txt”*

Removing and Preventing the Vulnerabilities Requires More Specific Definitions…



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

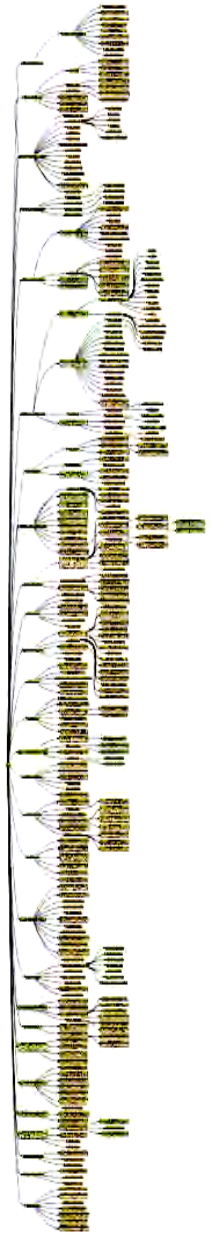
- Path Issue - dot dot slash - '../filedir'
- Path Issue - leading dot dot slash - '/../filedir'
- Path Issue - leading directory dot dot slash - '/directory/../filename'
- Path Issue - directory doubled dot dot slash - 'directory/../../filename'
- Path Issue - dot dot backslash - '..\filename'
- Path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

Vulnerability Theory: Problem Statement and Rationale

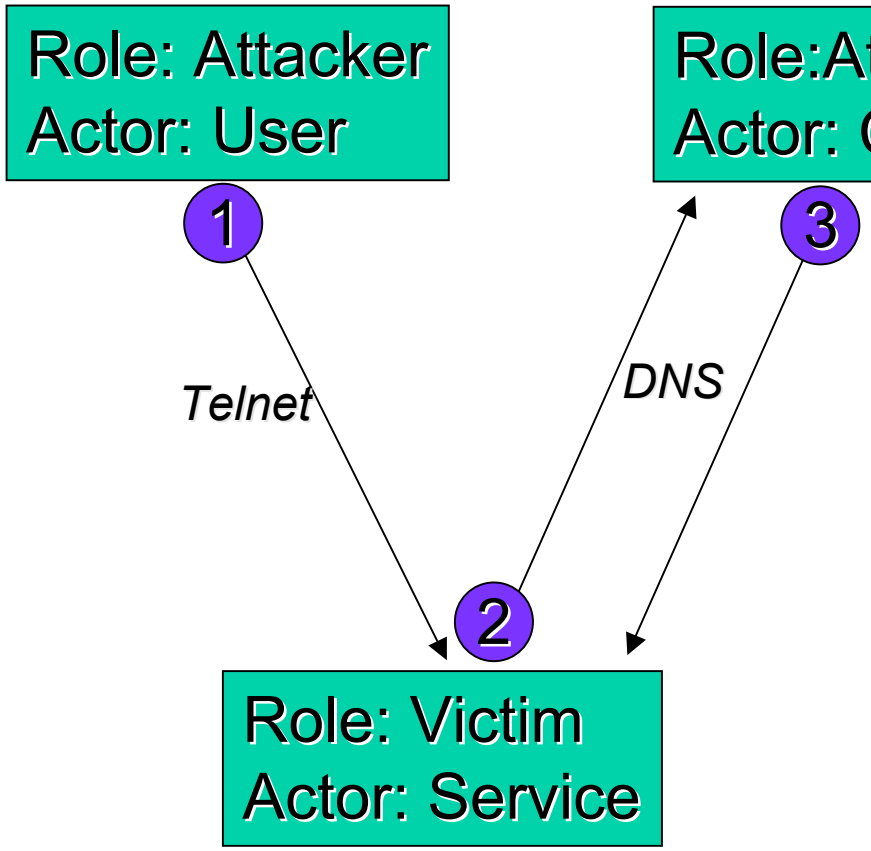
- With 600+ variants, what are the main themes?
- Why is it so hard to classify vulnerabilities cleanly?
  - CWE, Pernicious Kingdoms, OWASP, others have had similar difficulties
- Same terminology used in multiple dimensions
  - Frequent mix of attacks, threats, weaknesses/faults, consequences
  - E.g. 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, By Example

Buffer overflow using long DNS response



1) Attacker (as user) sends directive over Telnet channel: "Log me in"
2) Server (the target) sends directive over DNS channel: "Tell me IP's hostname"
3) DNS consultant (controlled by attacker) returns hostname with property ">300 BYTES"
4) Buffer overflow activated

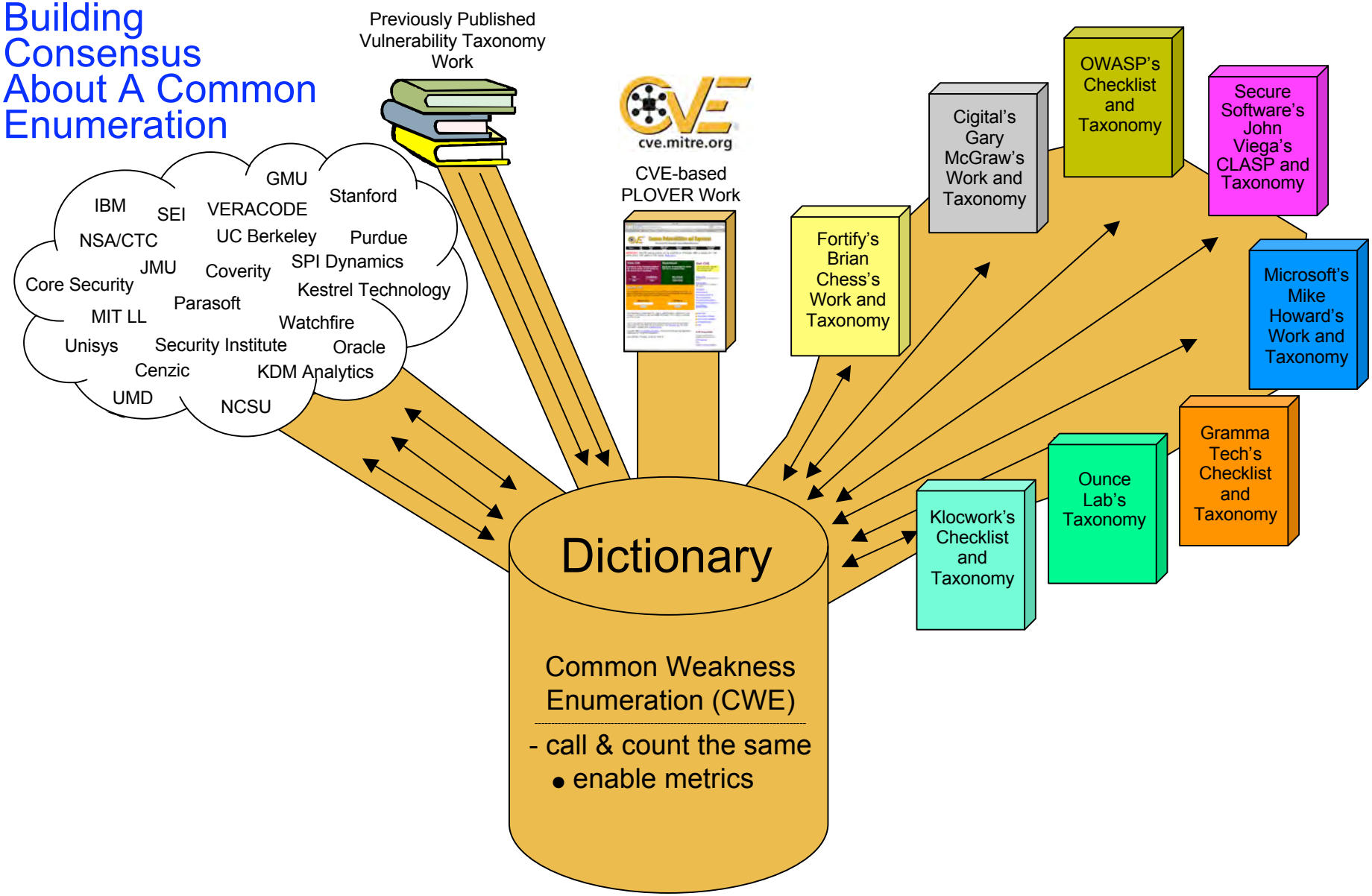
Artifact Labels

- Interaction Point
  - A relevant point within the product where a user interacts with the product
- Intermediate Fault
  - A behavior by the product that has not yet affected correctness, but will
- Control Transfer Point
  - The point where the program's behavior changes from correct to incorrect
- 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

Artifact Labels - Example

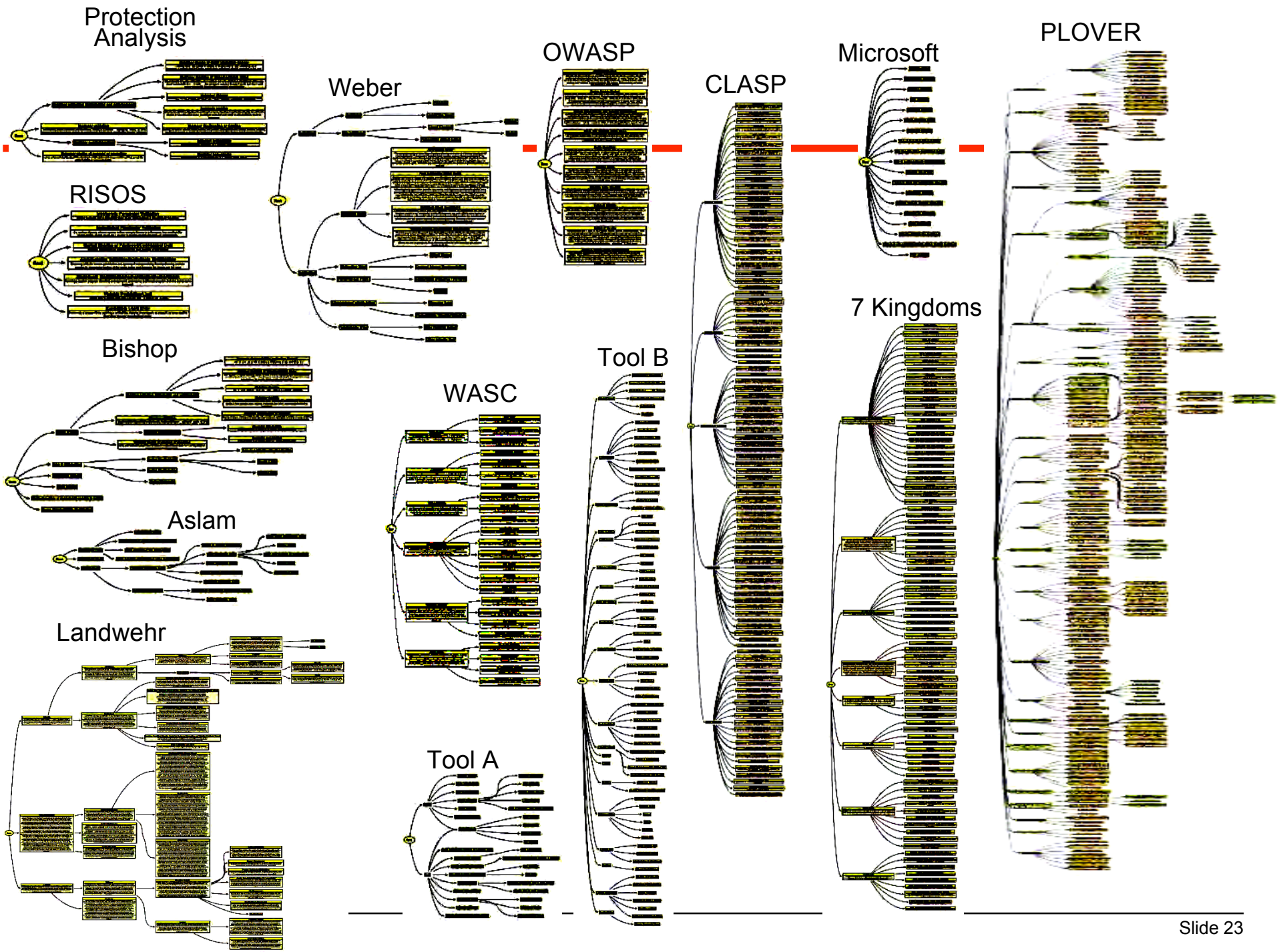
```
1     print HTTPresponseHeader;
2     print “<title>Hello World</title>”;
3     ftype = HTTP_Query_Param(“type”);
4     str = “/tmp”;
5     strcat(str, ftype); strcat(str, “.dat”);
6     handle = fileOpen(str, ”read”);
7     while((line=readFile(handle)))
8     {
9       line=stripTags(line, “script”);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

Building Consensus About A Common Enumeration



Where Did We Start?

- Objective: To identify, integrate and effectively describe common software weaknesses known to the industry and software assurance community
- Leveraging taxonometric 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


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

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 vulnerabili… cookies, create req… user, compromise … code on the end use…
Alternate Terms	"CSS" was once used … confusion with the "C… significantly, and its u…
Likelihood of Exploit	High to Very High
Weakness Ordinality	Resultant (Weakness … Weaknesses)
Causal Nature	Explicit (This is an exp… developer)
Common Consequences	Confidentiality: The m… scripting involves the … Access control: In som… code on a victim's com… other flaws
Potential Mitigations	Carefully check each i… specification (white lis… All input should be san… to specify, but all data… headers, the URL itsel… continuing XSS vulner…
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

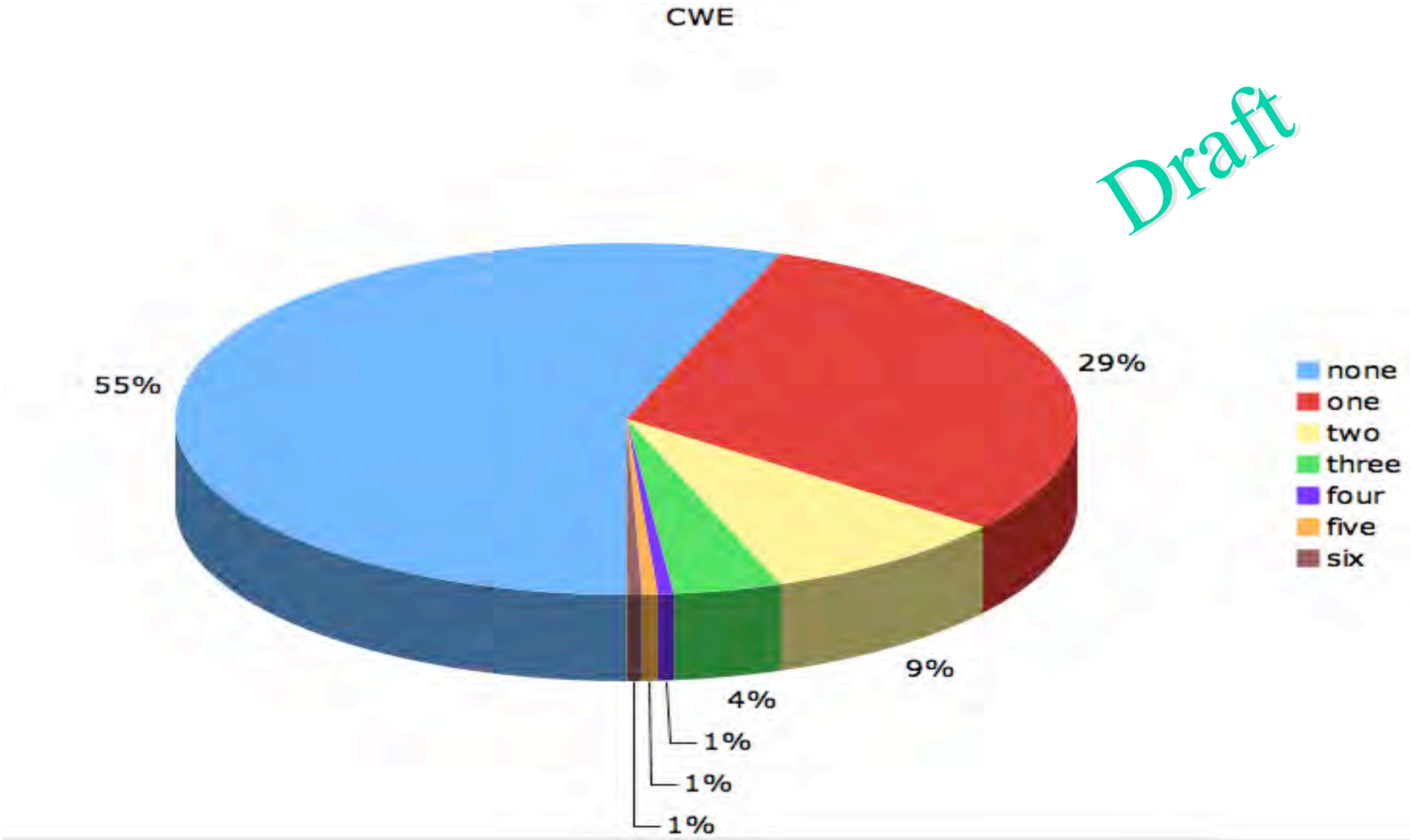
Accessibility
- Website is live with:
  - Historical materials, papers, alphabetical full enumeration, taxonomy HTML tree, CWE in XML, ability to URL reference individual CWEs, etc

Using A Unilateral NDA with MITRE to Bring in Info

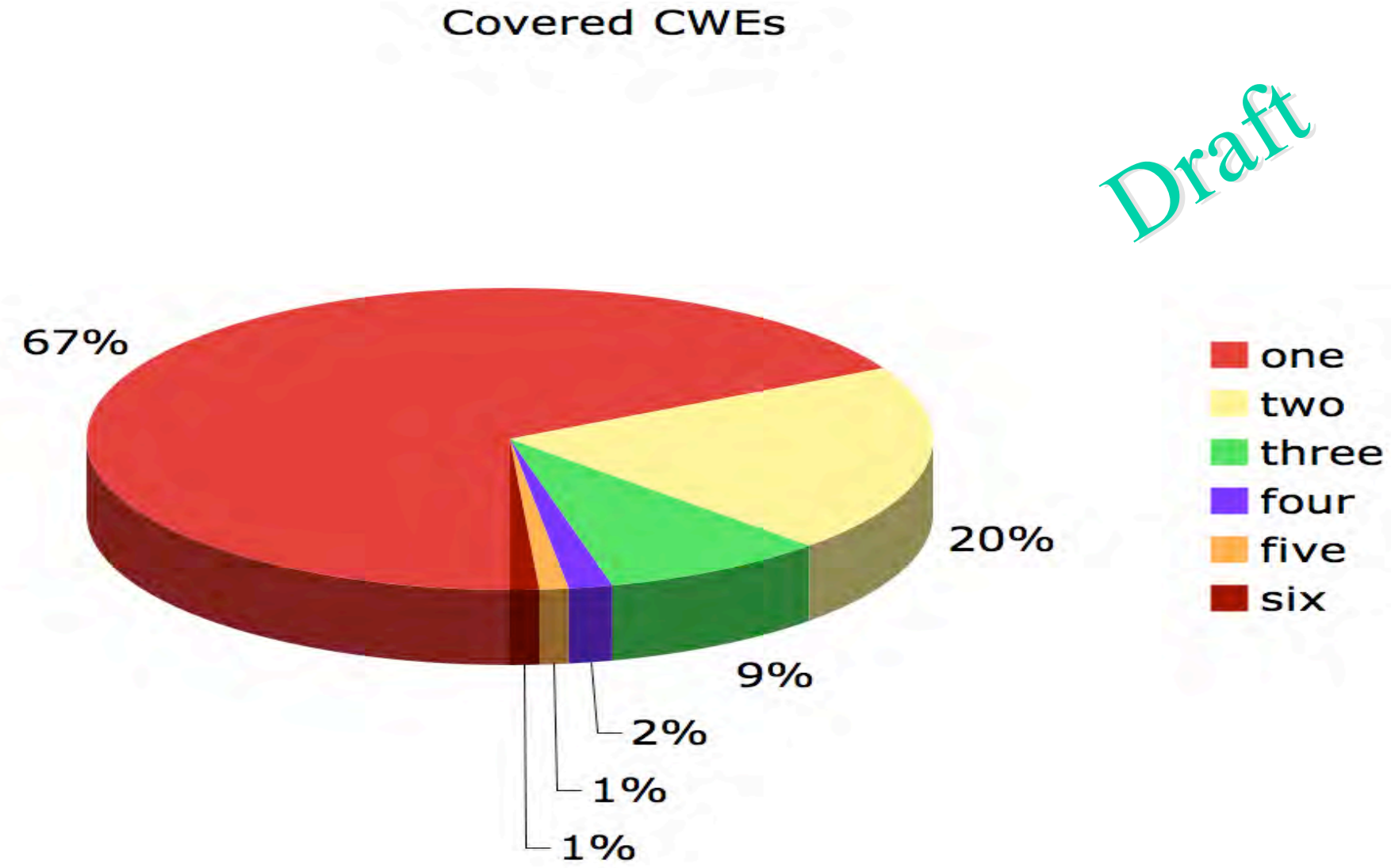
Purpose:

- 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.

Coverage of CWE



Covered CWEs - By Number of Tools



Current Community Contributing to the Common Weakness Enumeration

- AppSIC
- Cenzic
- 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
- UNISYS
- VERACODE
- Watchfire
- WASC
- Whitehat Security, Inc.
- Tim Newsham

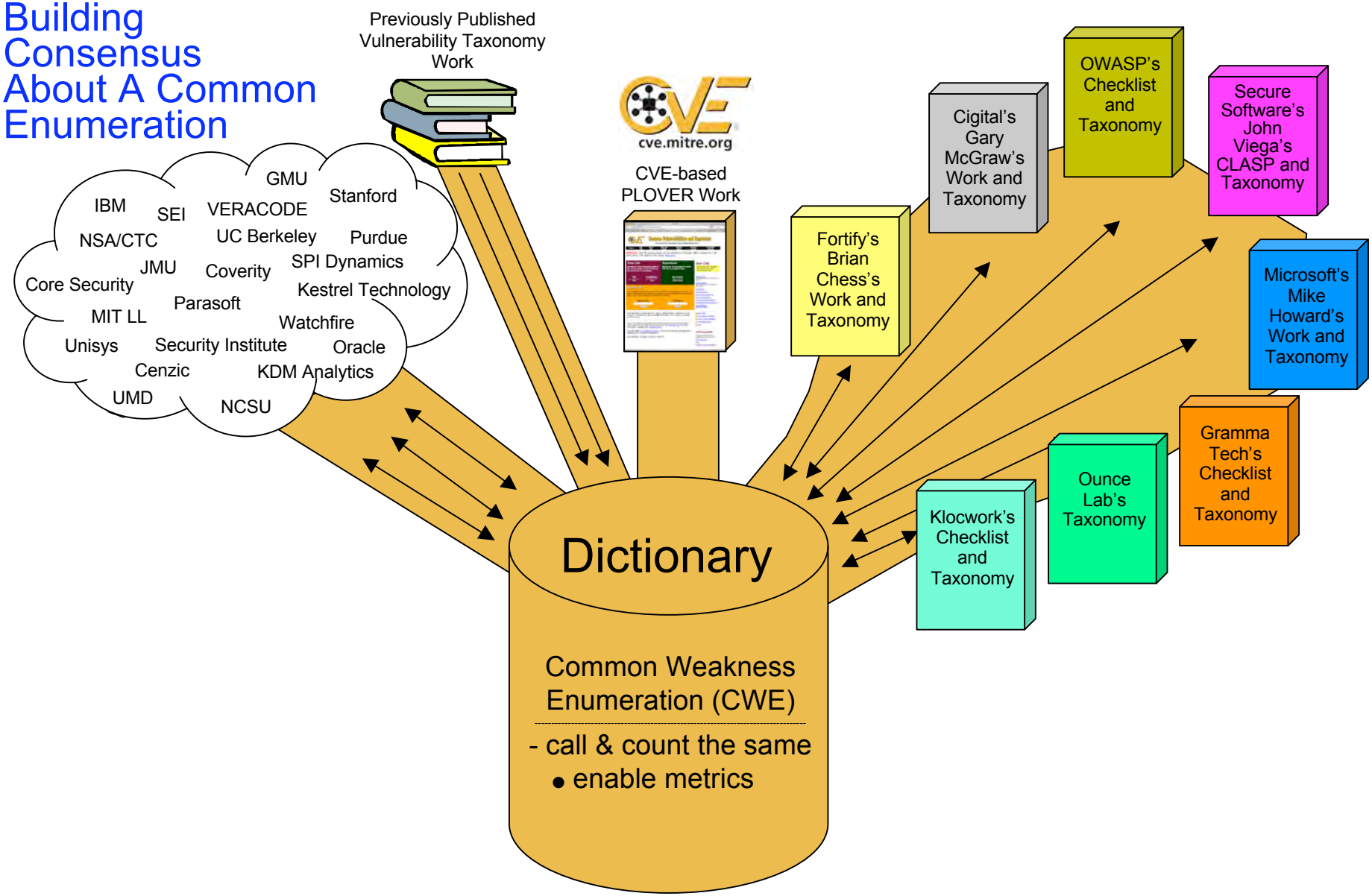
Planned Improvements - Content

- 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

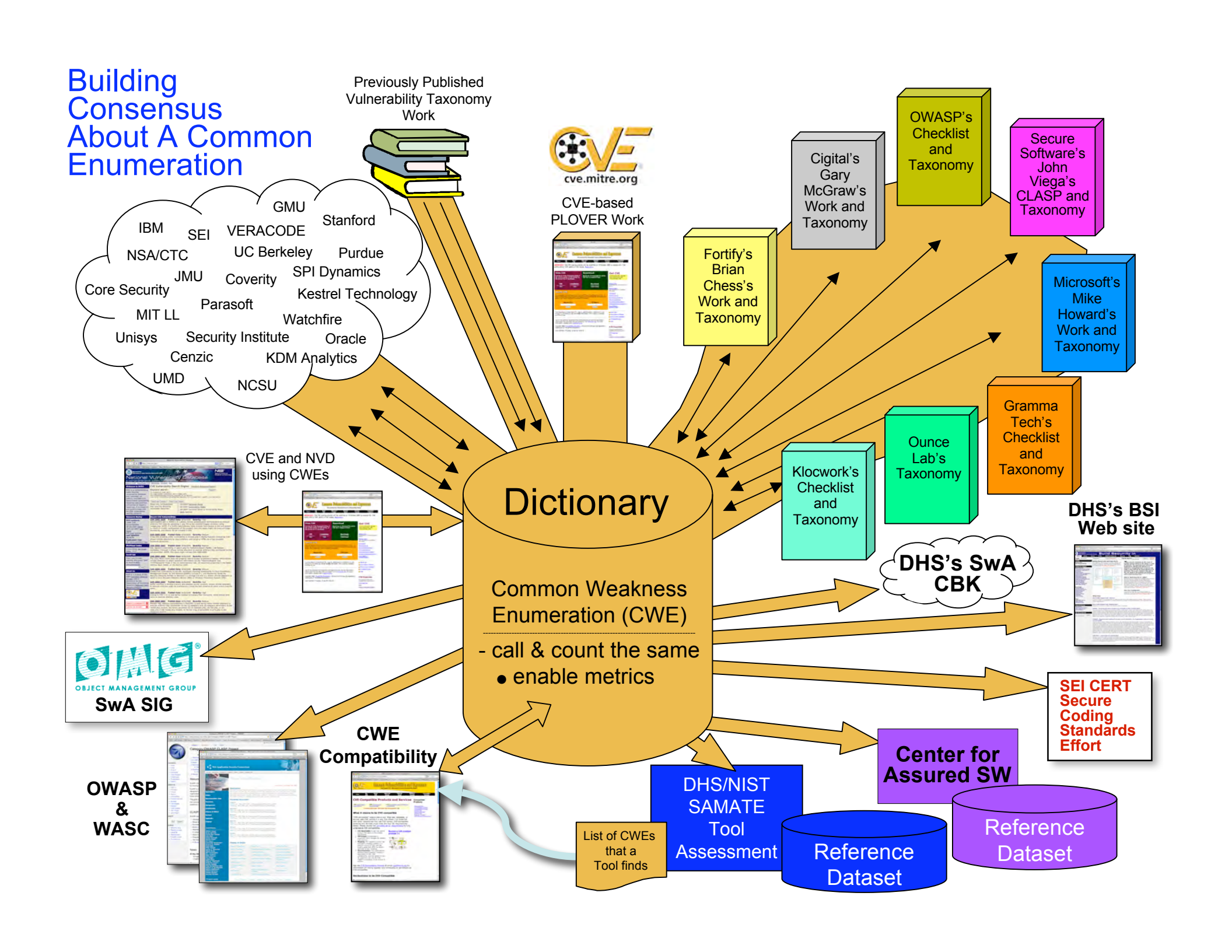
Planned Improvements - Site Usability

- 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

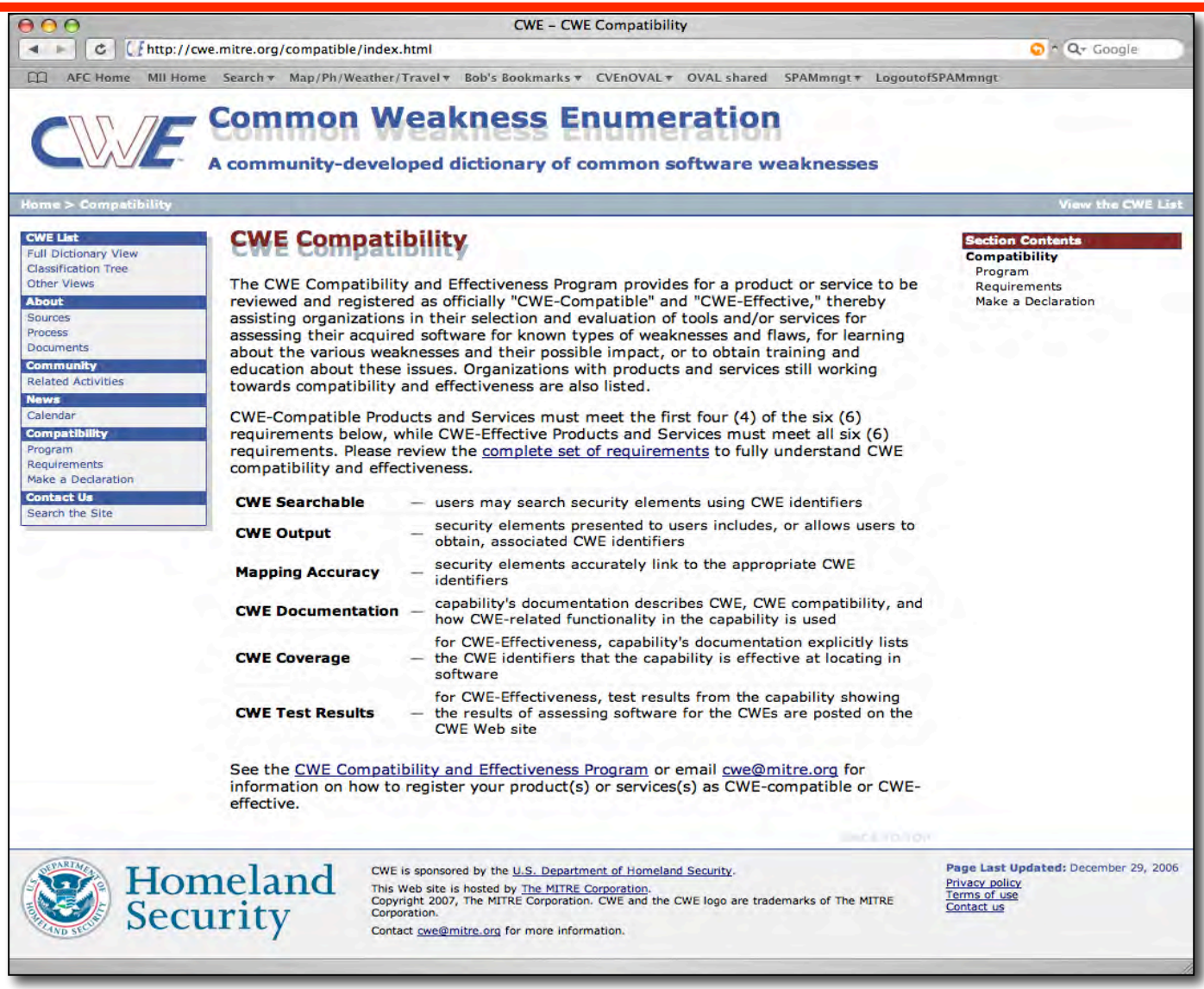
Building Consensus About A Common Enumeration



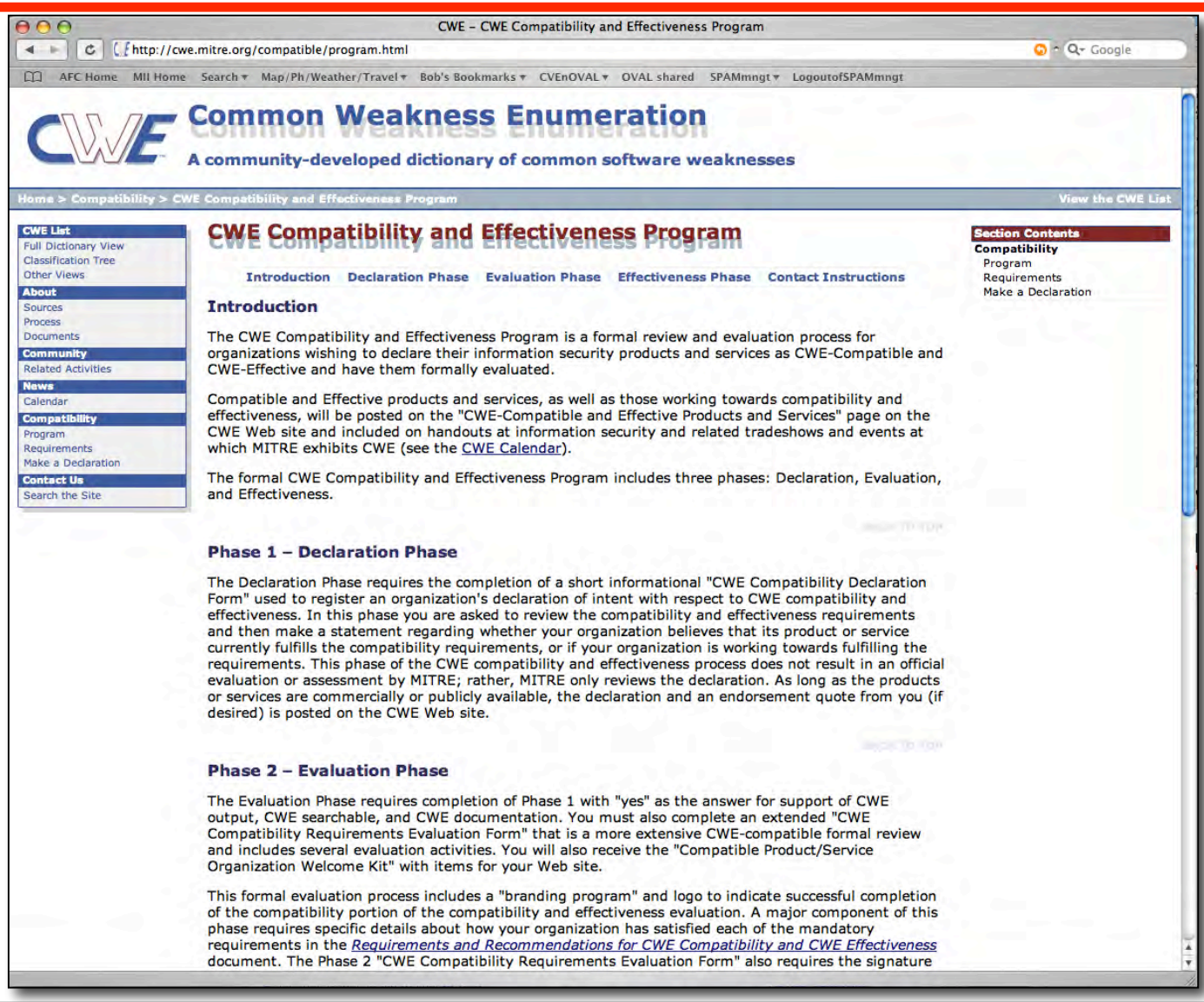
Building Consensus About A Common Enumeration



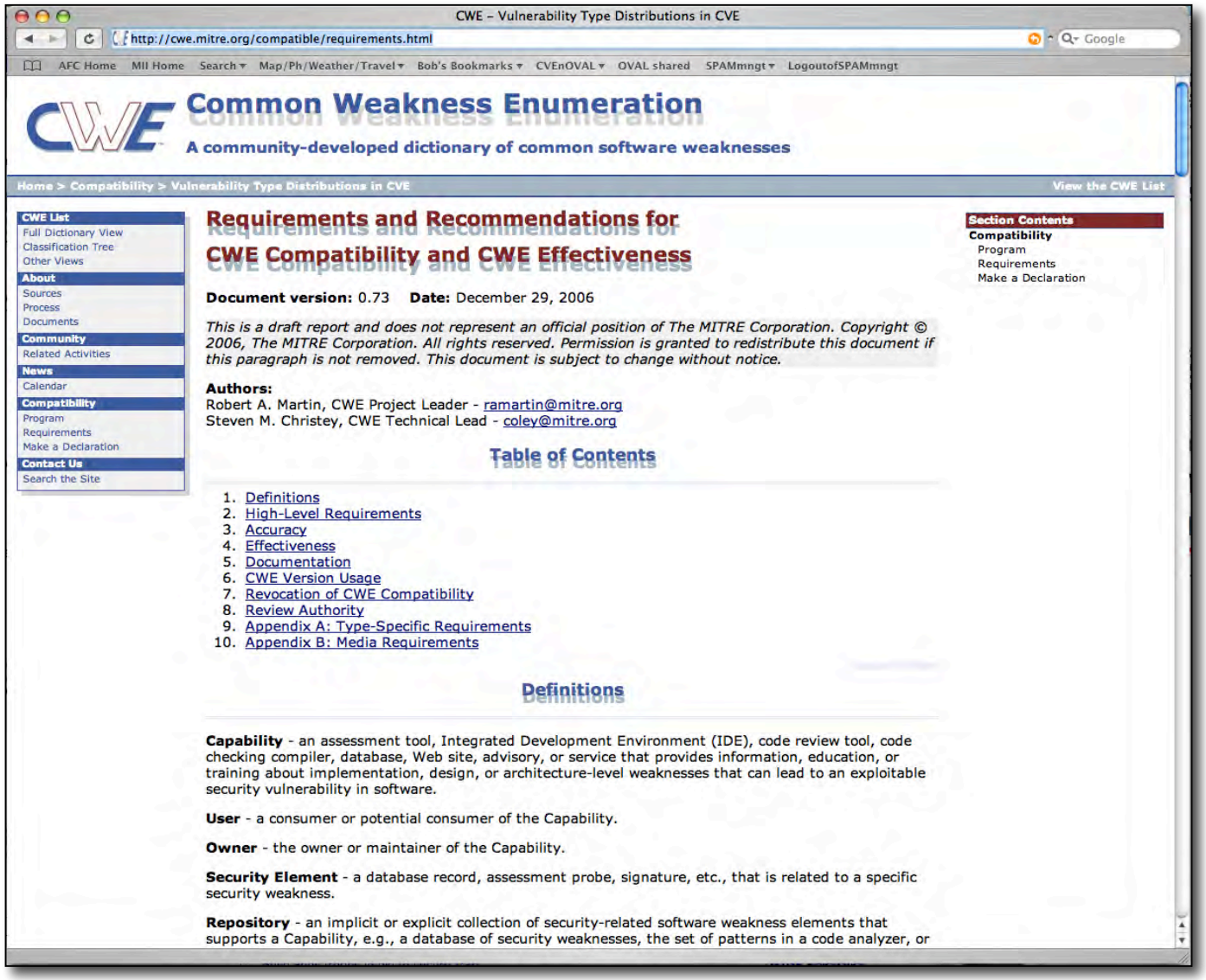
CWE Compatibility and Effectiveness Program Launched



CWE Compatibility and Effectiveness Process Posted



CWE Compatibility and Effectiveness Requirements Posted



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

The Road Ahead for the CWE effort

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