Having a Defined Target for Software Security Testing

Robert A. Martin

Making Security Measurable™

MITRE

The Primary Question of Testing is

What should I test?

- For traditional testing, process is well established (bounded by requirements)
- For security testing, the scope is not as broadly established

Making Security Measurable™

Flavors of Software Security Testing

- Functional Security Testing
  - **Test the security-related features of the system**
  - **Ensure they behave in the prescribed manner (e.g., login features)**
- Risk-Based Security Testing
  - **Ensure secure behavior of "other" requirements**
  - **Testing to assure absence of vulnerabilities**
  - **Testing negative requirements (misuse and abuse cases)**
    - Assure resilience to likely attacks
  - **Ensure security risks introduced during software development have been effectively mitigated**

Making Security Measurable™

Creating a Testing Target List

- Functional security testing is the same as traditional testing
  - **Testing target list driven by requirements**
- Testing to assure absence of vulnerability
  - **Need a resource for what weaknesses/vulnerabilities to test for**
- Testing to assure resistance/resilience to attack
  - **Need a resource for what attacks to simulate for resistance/resilience testing**

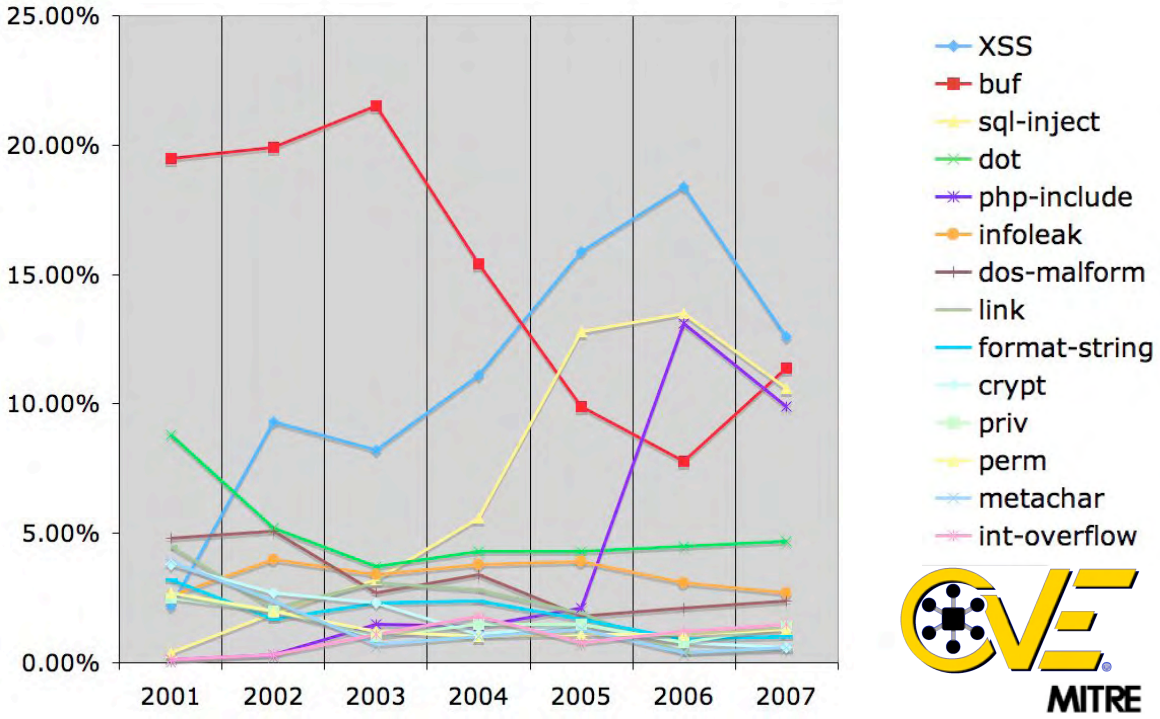
Making Security Measurable™

© 2008 MITRE

A Resource for Creating the Weakness/Vulnerability Testing Target List

Making Security Measurable™

Vulnerability Type Trends: A Look at the CVE List (2001 - 2007)



CVE MITRE

XSS Attack Modes

- Attack modes
  - **Send a link that the victim clicks on**
  - **Send an email containing script that automatically "clicks" the link for the user**
    - Javascript / AJAX / Flash / etc. can be very powerful
  - **Convince a user to visit a web page**
    - Or compromise a web page yourself
- Typical consequences
  - **Steal cookies, e.g. those used for authentication**
  - **Phishing - redirect user to malicious site**
  - **Malware - exploit client-side vulnerabilities**
  - **Modify content – fake news stories or stock prices anyone?**
- While XSS is most frequent in web applications, it can also happen to web server software (e.g. IIS, Apache), web browsers (e.g. Internet Explorer, Firefox), and web-friendly desktop applications (e.g. Flash, PDF)

Making Security Measurable™

XSS is possible anywhere scripting is supported

- <script …>
- <img src=javascript:alert('XSS')>
- CSS styles
- <b> tags
- DOM-based XSS
  - **Client-side Javascript doesn't handle inputs properly**
- Direct insertion of javascript into code segment of web page
- Flash, PDF, other web-friendly technologies
- Encoding
- Filter bypass

The wide variety of attacks and weaknesses is one reason why XSS is so common… and why CVE descriptions try to list the variants unless they're <SCRIPT> or <img src=javascript> examples.

Making Security Measurable™

Javascript Splicing

- Another XSS variant, but into javascript portion of the generated web page
  - **aka "Javascript injection"**
- Filtering <script> etc. is not effective
  - **Injections follow Javascript syntax, so "()" and ";" become relevant**
- Example: CVE-2007-2581

Making Security Measurable™

Blacklists and XSS

- "I'll just strip uses of <SCRIPT>"
  - **This works: <img src="javascript:alert('hi')">**
- "But I want to support IMG tags, so I'll just strip 'javascript'"
  - **This works (no lie): <img src="j&#X41vascript:alert('hi')">**
- "I'll decode everything, THEN look for 'javascript'"
  - **This works (no lie): <img src="javas cript:alert('hi')**
- "I'll make sure that only 'http' is allowed"
  - **This works: <img src="http://www.example.com/pic.jpg" onmouseover="alert('hi')">**
- "I'll make sure to strip out 'onmouseover'"
  - **This works: <img src="http://www.example.com/pic.jpg" onload="alert('hi')">**
- "I'll only support SRC for IMG tags"
  - **This works (no lie): hello**
- … and many, many more

Making Security Measurable™

Insufficient protection schemes often affect the exact same vector in multiple CVE's. One CVE for the original missing XSS protection, another CVE for the wrong protection.

Integer Overflows

- 32bit vs. 64bit
- #2 issue for OSes
- Not just for C

- Frequently bypassed in signed comparison

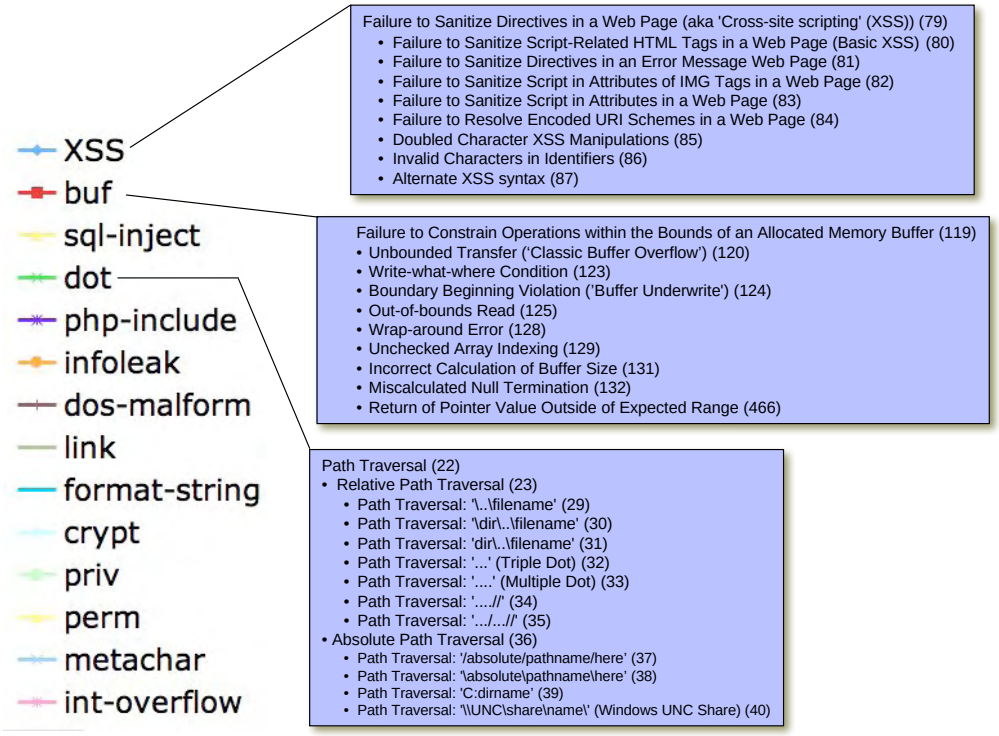
Making Security Measurable™

Buffer Overflow Variants

- Array index errors
  - **Attacker can control the index of an array**
  - **Often allows writing to / reading from arbitrary locations**
  - **Can apply to any language that indexes arrays**
    - Negative arguments have special meaning in Perl
    - Large indices could trigger creation of large data structure
- Closely related: user-controlled offsets
  - **Found in callback management routines**
  - **Found in plugin capabilities**

Making Security Measurable™

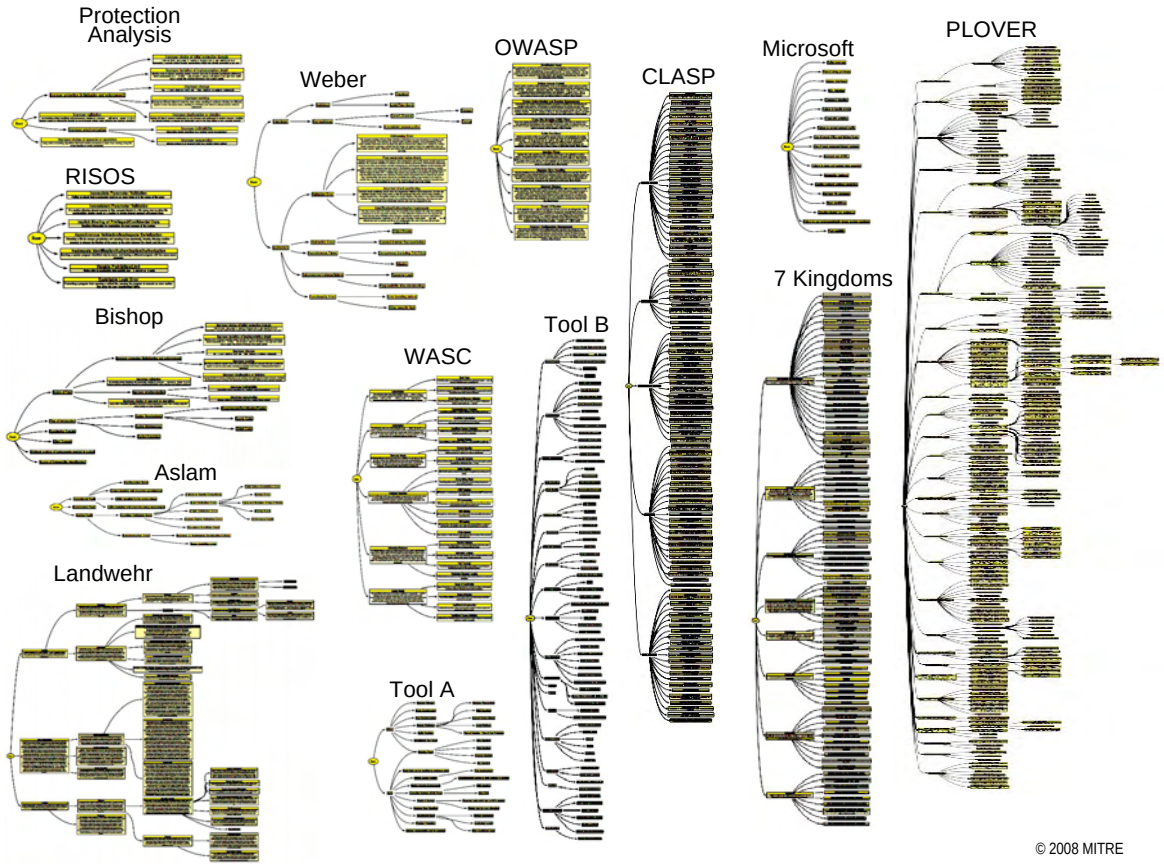
Removing and Preventing the Vulnerabilities Requires More Specific Definitions…CWEs



Goal of the Common Weakness Enumeration Initiative

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

Making Security Measurable™

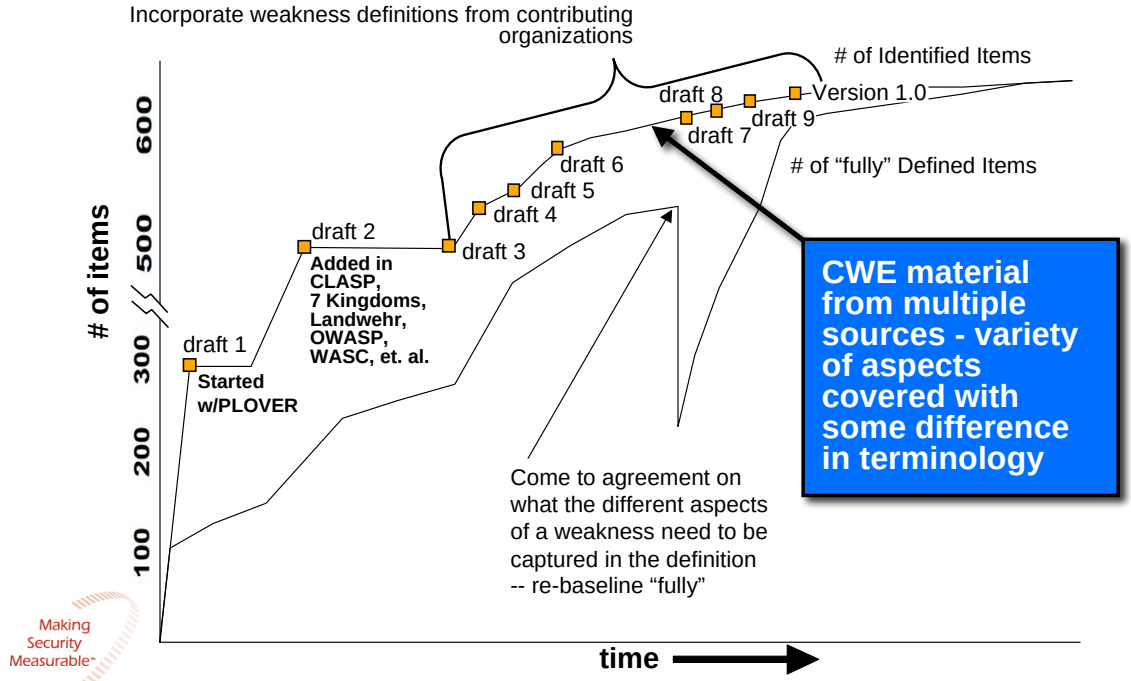


Using A Unilateral NDA with MITRE to Bring in Info

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.

Timeline of Items Enumerated and Defined in CWE



Making Security Measurable™

Formalizing a Schema for Weaknesses

Identifying Information
- CWE ID
- Name

Describing Information
- Description
- ***Extended Description***
- Alternate Terms
- Demonstrative Examples
- Observed Examples
- Context Notes
- Source Taxonomy
- References
- ***Whitebox Definition***
- ***Blackbox Definition***
- ***Formal Definition***

Scoping & Delimiting Information
- Type
- 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
- Affected Resource
- ***Related Attacks***
- ***Detection Factors***
- Node Relationships
- Research Gaps

Making Security Measurable™

CWE Content Fields Defined

http://cwe.mitre.org/documents/schema/index.html

Schema Documentation

Document version: 0.5 **Date:** April 9, 2008

This is draft document. It is intended to support maintenance of CWE, and to educate and solicit feedback from a specific technical audience. This document does not reflect any official position of the MITRE Corporation or its sponsors. Copyright © 2008, The MITRE Corporation. All rights reserved. Permission is granted to redistribute this document if this paragraph is not removed. This document is subject to change without notice.

Author: Conor Harris
URL: http://cwe.mitre.org/documents/schema/index.html

Table of Selected Content

Affected_Resource	Alternate_Terms	Applicable_Platforms	Black_Box_Definition	CAPEC_ID
Catalog_Name	Catalog_Version	Categories	Category_ID	Category_Name
Category_Status	Causal_Nature	Code_Block	Code_Example_Language	Common_Consequence
Common_Consequences	Compound_Element_Abstraction	Compound_Element_ID	Compound_Element_Name	Compound_Element_Status
Compound_Element_Structure	Compound_Elements	Context_Notes	Demonstrative_Example	Demonstrative_Example_Reference
Description	Description_Summary	Detection_Factor	Enabling_Factors_for_Exploitation	Example_Block
Example_Code	Example_Code_Block	Extended_Description	Functional_Area	Likelihood_of_Exploit
Mapped_Node_Name	Mapped_Taxonomy_Name	Mitigation	Modification	Modification_Comment
Modification_Date	Modification_Type	Modifier	Modifier_Organization	Observed_Example
Observed_Example_Description	Observed_Example_Link	Observed_Example_Reference	Observed_Examples	Ordinal
Original_Node_Name	Platform	PostText	Post_Code_Comment	Potential_Mitigations
PreText	Pre_Code_Comment	Reference	Reference_Author	Reference_Date
Reference_Edition	Reference_ID	Reference_Link	Reference_PubDate	Reference_Publication
Reference_Publisher	Reference_Section	Reference_Title	References	Related_Attack_Pattern
Related_Attack_Patterns	Relationship_Chain_ID	Relationship_Chains	Relationship_Nature	Relationship_Target_ID
Relationship_Type	Relationship_View_ID	Relationship_Views	Relationships	Relevant_Properties
Relevant_Property	Research_Gaps	Source	Source_Taxonomy	Source_Taxonomy_Name
Submission	Submission_Comment	Submission_Date	Submission_Type	Submitter
Submitter_Organization	Taxonomy_Mapping	Time_of_Introduction	View_Filter	View_ID
View_Name	View_Status	Views	Weakness_Abstraction	Weakness_Catalog
Weakness_ID	Weakness_Name	Weakness_Ordinality	Weakness_Status	Weaknesses
White_Box_Definition				

Weakness_Catalog

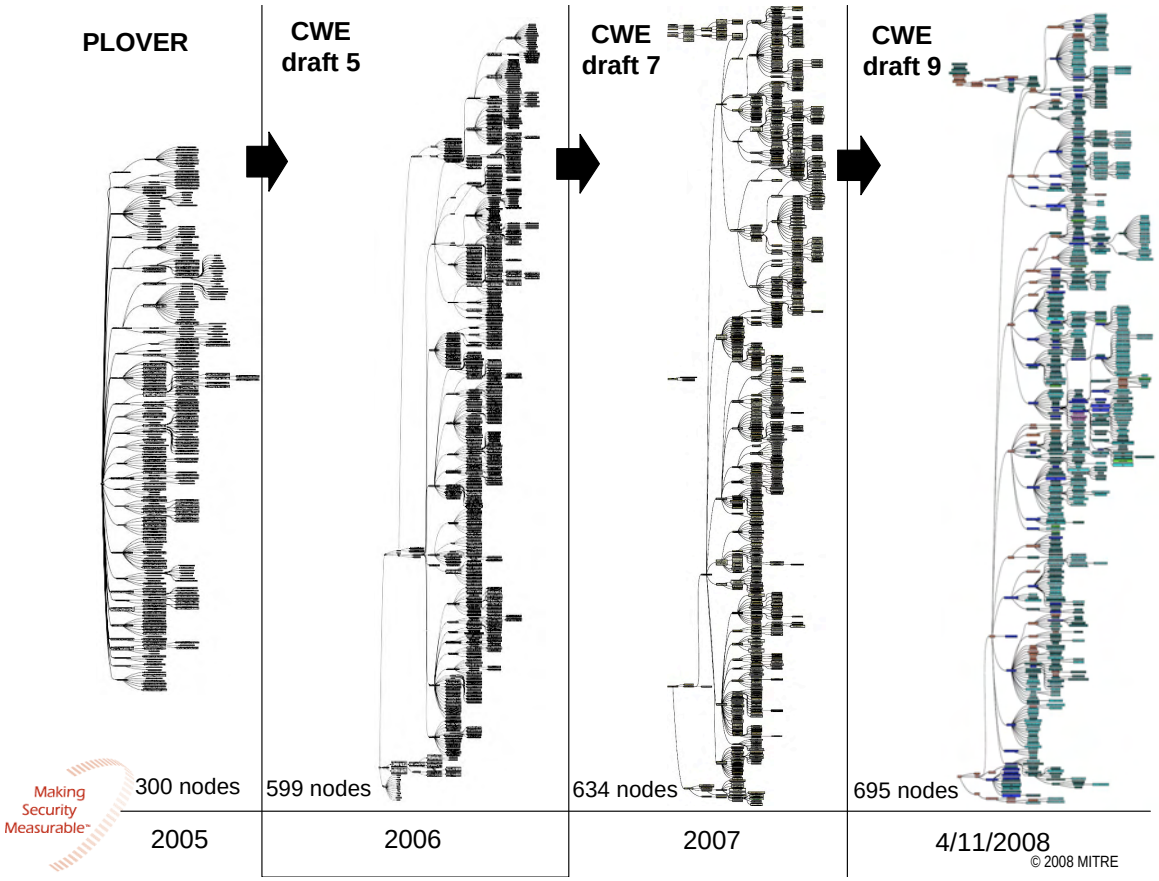
The Weakness_Catalog structure represents a collection of software security issues (flaws, faults, bugs, vulnerabilities, weaknesses, whatever). The name used by CWE is usually "CWEC", however if this collection is a subset of CWE then a more appropriate name should be used. It has two required attributes: Catalog_Name and Catalog_Version

Views

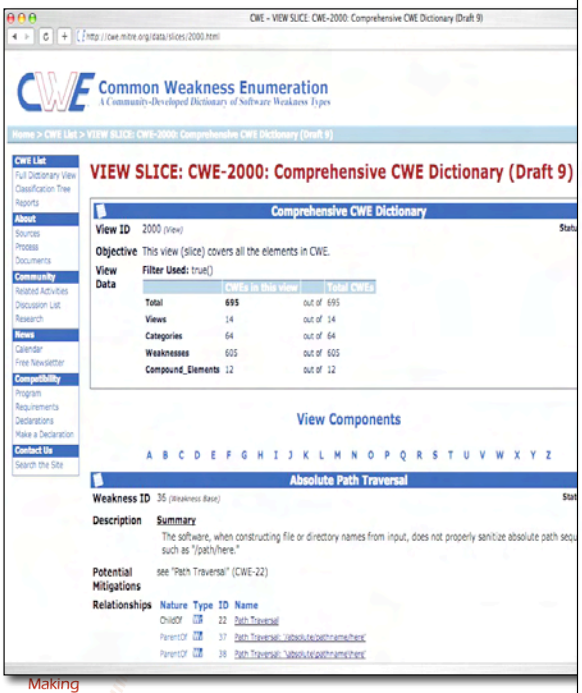
The Views structure contains zero or more View elements. Each View element represents a perspective with which one might look at the weaknesses in CWE. CWE-630 Weaknesses Examined by SAMATE and CWE-658 Weaknesses found in the C Language are two examples of Views.

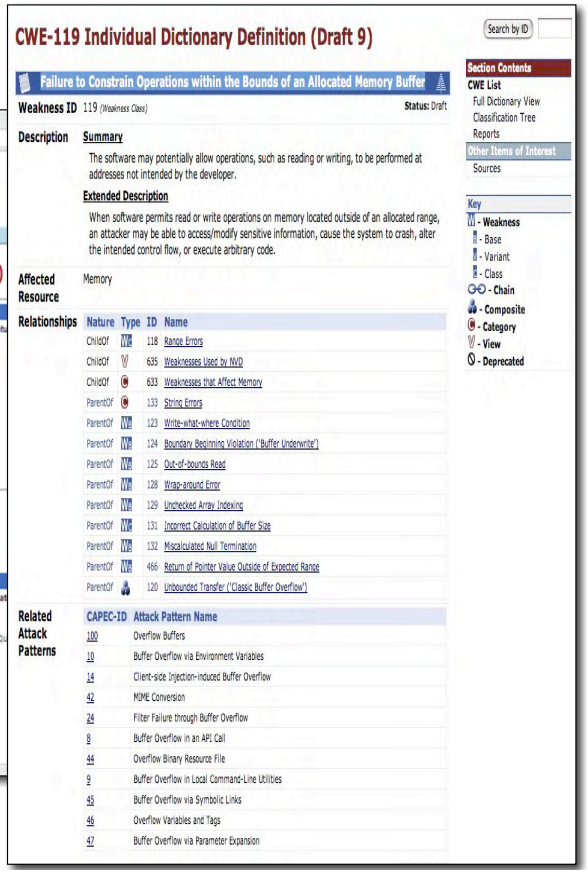
Categories

The Categories structure contains zero or more Category elements. Each Category element represents what used to be referred to in CWE as a "Grouping" entry. That is, a category is now a collection of weaknesses based on a common attribute, such as CWE-310 Cryptographic Issues or CWE-355 User Interface Security Issues.



CWE Draft 9 (11 April 08)



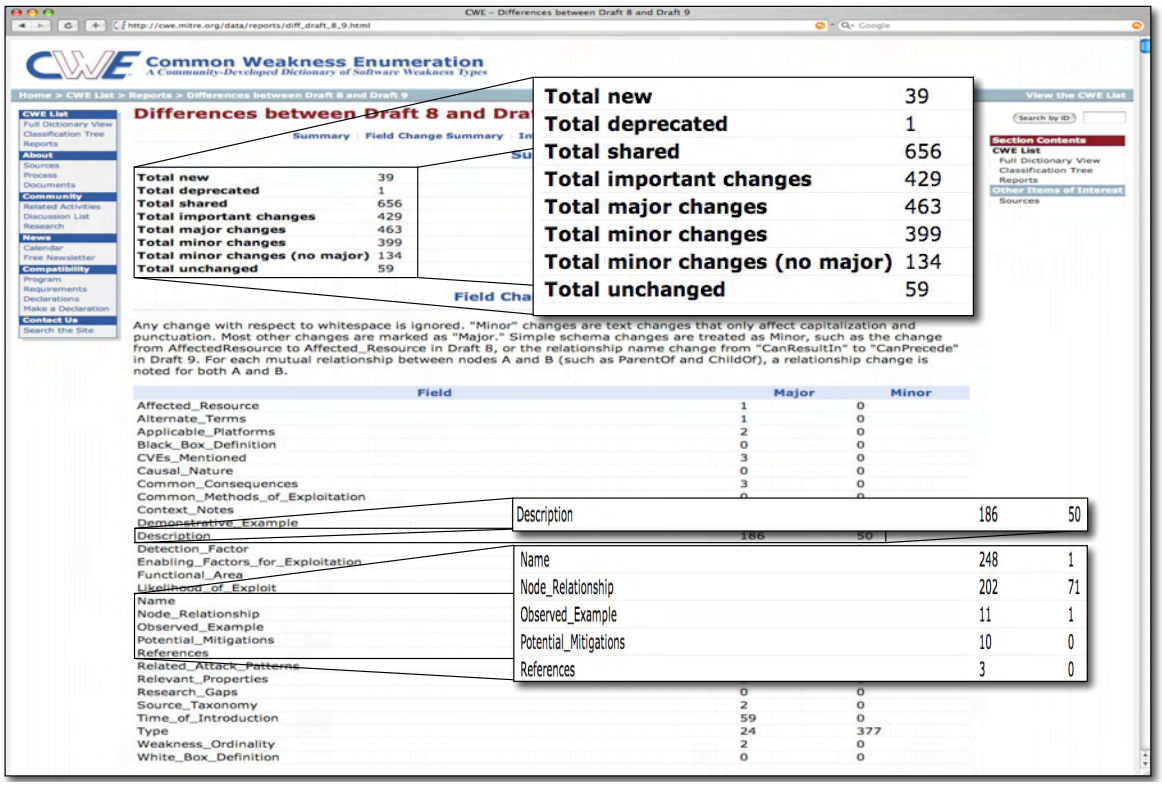


Making Security Measurable™

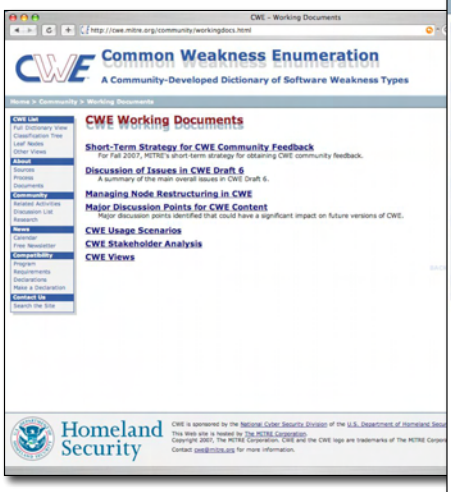
The Classification Problem: Same Term, Many Perspectives, Lots of Overlap

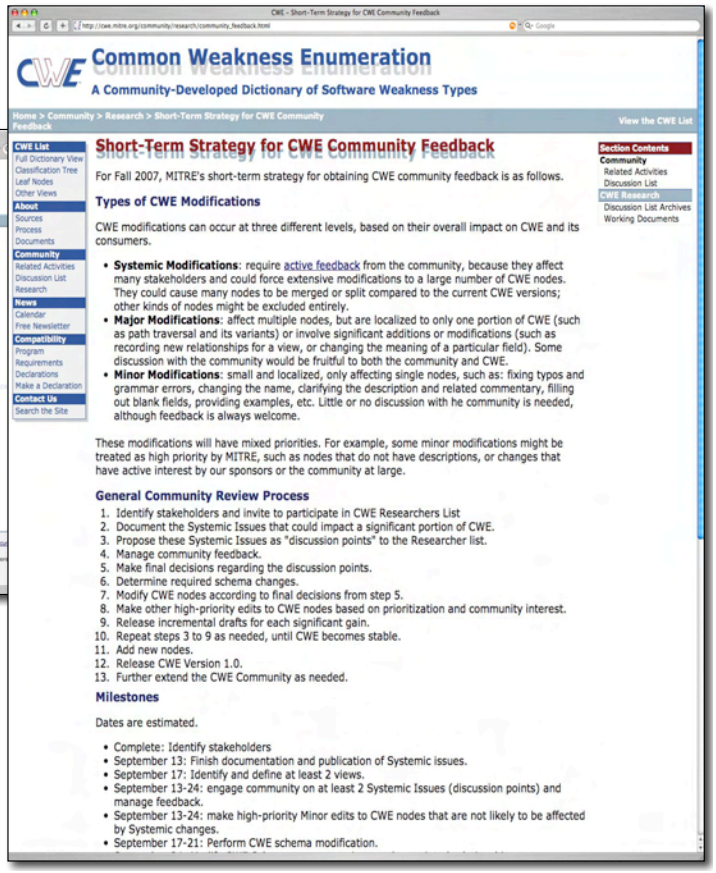
Term	Attack	Vuln/Weakness	Consequence
Buffer Overflow	•Long string argument •Length field inconsistency •Large number of events, etc.	•Failure to restrict length •Failure to control offset •Error in attempting to do either of these	•Write of data past explicitly specified boundaries of a buffer •Crash, code execution, control/data flow modification
Format String	Format string specifiers relative to the underlying representation in use (typically C-style strings)	•Failure to fully control contents of format strings	•Write of data past explicitly specified boundaries of a buffer •Crash, code execution, control/data flow modification
Directory Traversal	"..", "/a/b/c", "..../", etc.	Failure to properly restrict file within intended subdirectory	Access of file outside intended subdirectory
Information Leak	•Provide invalid argument •Monitor behavioral or timing results •Sniff	•Failure to anticipate error conditions •Failure to limit info in error messages •Failure to zero out sensitive info	Disclosure of sensitive information relative to an implicit or explicit policy of what constitutes "sensitive"
XSS	•<SCRIPT>alert('hi')</SCRIPT> •"javascript:alert(document.cookie)" •"java#42;script:abc" ...	Failure to properly filter, escape, or encode outputs with respect to their particular role (e.g. tags or tag arguments), in a fashion that is syntactically or semantically valid for the representation and encoding that are currently in use	•Execution of script code •Modification of format or presentation
DoS	Provide invalid argument	•Failure to anticipate or handle error conditions •Failure to properly limit scope of an error	•Crash •"Memory Corruption" •Infinite loop
DoS	•Large number of events •Send a large amount of data •Manipulate algorithmic complexity	Failure to sufficiently control resource consumption relative to performance expectations for the application and/or its environment	•Crash •"Memory Corruption" •Infinite loop
Authentication Bypass	•Perform invalid sequence of instructions, e.g. direct request •Replay challenge/response •Cookie modification •SQL injection	•Failure to enforce required sequence of steps •Failure to prevent modification of assumed-immutable data •Secondary effect of primary issue	Access privileged functionality or data before fully navigating all required authentication steps

Draft 8 to Draft 9 Changes



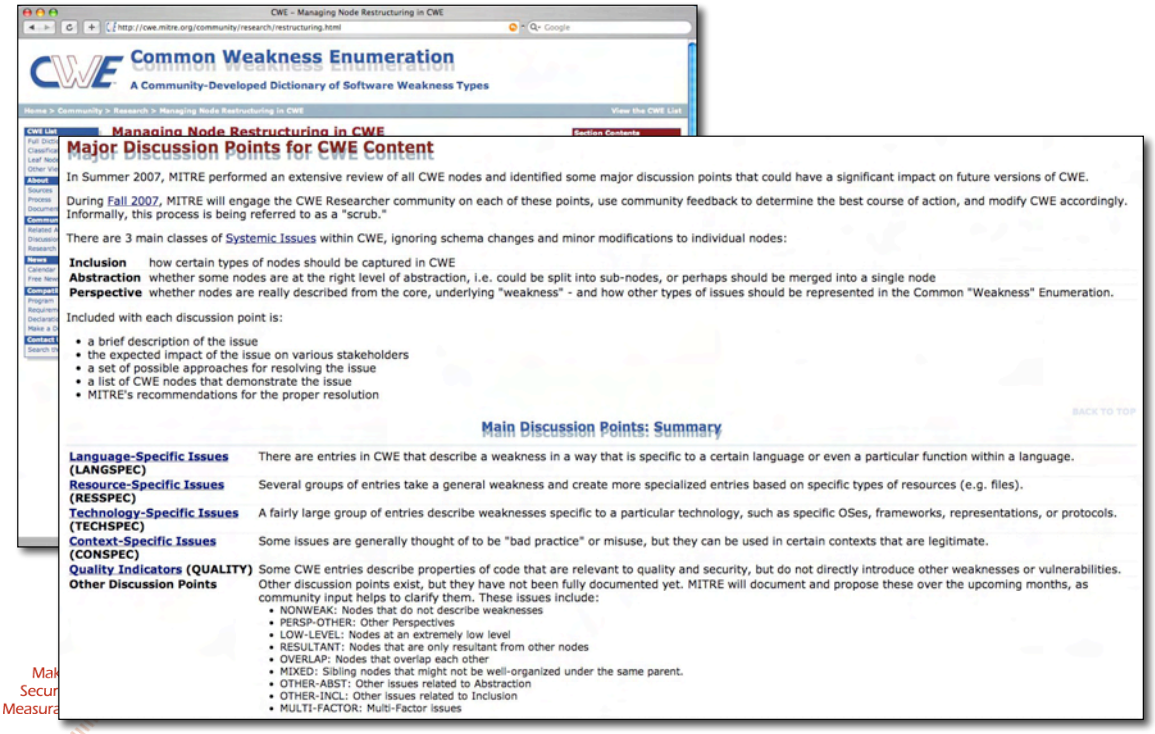
CWE "Scrub"





Making Security Measurable™

CWE "Scrub"



CWE - Managing Node Restructuring in CWE

http://cwe.mitre.org/community/research/restructuring.html

Common Weakness Enumeration

A Community-Developed Dictionary of Software Weakness Types

Home > Community > Research > Managing Node Restructuring in CWE

View the CWE List

Managing Node Restructuring in CWE

Major Discussion Points for CWE Content

In Summer 2007, MITRE performed an extensive review of all CWE nodes and identified some major discussion points that could have a significant impact on future versions of CWE.

During Fall 2007, MITRE will engage the CWE Researcher community on each of these points, use community feedback to determine the best course of action, and modify CWE accordingly. Informally, this process is being referred to as a "scrub."

There are 3 main classes of Systemic Issues within CWE, ignoring schema changes and minor modifications to individual nodes:

Inclusion how certain types of nodes should be captured in CWE
Abstraction whether some nodes are at the right level of abstraction, i.e. could be split into sub-nodes, or perhaps should be merged into a single node
Perspective whether nodes are really described from the core, underlying "weakness" - and how other types of issues should be represented in the Common "Weakness" Enumeration.

Included with each discussion point is:

- a brief description of the issue
- the expected impact of the issue on various stakeholders
- a set of possible approaches for resolving the issue
- a list of CWE nodes that demonstrate the issue
- MITRE's recommendations for the proper resolution

BACK TO TOP

Main Discussion Points: Summary

Language-Specific Issues (LANGSPEC)	There are entries in CWE that describe a weakness in a way that is specific to a certain language or even a particular function within a language.
Resource-Specific Issues (RESSPEC)	Several groups of entries take a general weakness and create more specialized entries based on specific types of resources (e.g. files).
Technology-Specific Issues (TECHSPEC)	A fairly large group of entries describe weaknesses specific to a particular technology, such as specific OSes, frameworks, representations, or protocols.
Context-Specific Issues (CONSPEC)	Some issues are generally thought of to be "bad practice" or misuse, but they can be used in certain contexts that are legitimate.
Quality Indicators (QUALITY)	Some CWE entries describe properties of code that are relevant to quality and security, but do not directly introduce other weaknesses or vulnerabilities.
Other Discussion Points	Other discussion points exist, but they have not been fully documented yet. MITRE will document and propose these over the upcoming months, as community input helps to clarify them. These issues include:

- NONWEAK: Nodes that do not describe weaknesses
- PERSP-OTHER: Other Perspectives
- LOW-LEVEL: Nodes at an extremely low level
- RESULTANT: Nodes that are only resultant from other nodes
- OVERLAP: Nodes that overlap each other
- MIXED: Sibling nodes that might not be well-organized under the same parent.
- OTHER-ABST: Other issues related to Abstraction
- OTHER-INCL: Other issues related to Inclusion
- MULTI-FACTOR: Multi-Factor issues

Mak
Secur
Measura

CWE Stakeholders

- Assessment Vendors
- Assessment Customers
- Software Developers
- Software Customers
- Academic Researchers
- Applied Vulnerability Researchers
- Refined Vulnerability Information Providers
- Educators
- Specialized Communities:
  - Web **application security community (e.g. WASC, OWASP)**
  - **NIST SAMATE**
    - want to understand tool capabilities
  - **Secure code development**
  - **Secure coding standards**
    - encourages the adoption of coding practices to avoid vulnerabilities (e.g. the CERT Secure Coding Standards project).
  - **Language vulnerability avoidance**
    - provide guidance to programmers in avoiding vulnerabilities inherent in programming languages and guidance to language developers in improving their language standards (e.g. ISO/IEC TR 24772 being developed by ISO/IEC JTC 1/SC 22/OWGV)

Making Security Measurable™

CWE – CWE Stakeholder Analysis

http://cwe.mitre.org/community/research/stakeholders.html

Common Weakness Enumeration

A Community-Developed Dictionary of Software Weakness Types

Home > Community > Research > CWE Stakeholder Analysis | View the CWE List

CWE List
Full Dictionary View
Classification Tree
Leaf Nodes
Other Views
About
Sources
Process
Documents
Community
Related Activities
Discussion List
Research
News
Calendar
Free Newsletter
Compatibility
Program
Requirements
Declarations
Make a Declaration
Contact Us
Search the Site

CWE Stakeholder Analysis

CWE Usage Scenarios | Stakeholder Analysis | Views

Section Contents
Community
Related Activities
Discussion List
CWE Research
Discussion List Archives
Working Documents

Assessment Vendors	Developers of code scanners, services, and other types of assessment technologies.

Priorities: Want their capabilities to be as comprehensive as possible while minimizing false positives/negatives. Want to market their strengths relative to competitors and identify their own limitations.

Challenges: how to prioritize enhancements; how to extend their capabilities quickly; how to present results; customers who don't know how to ask for what they want.

Dependencies: Academic Researchers, Software Developers, Applied Vulnerability Researchers, Specialized Communities.

Tasks:

1. Improve coverage (breadth or depth)
2. Support comparison/procurement
3. Provide correlating identifiers for customers: (a) to find more information, (b) to exchange information
4. Counting metric (# found vs. # tested)
5. Reduce false-positives/false-negatives
6. Discover weakness chains that have security impact
7. Map test cases to CWEs

Usage Scenarios:

- Mapping
- Find Related
- Find Gaps

Assessment Customers	Purchasers and users of assessment technologies and services, as provided by Assessment Vendors. These purchasers could be any of the other stakeholders in this list, especially Software Developers and Applied Vulnerability Researchers.

Priorities: want to find the right assessment capability for their needs; want sufficient documentation to understand the problems and devise strategies to fix them; want to know which issues aren't detected; want to minimize false positives/negatives.

Challenges:

Dependencies: Assessment Vendors, Specialized Communities.

Tasks:

Making Security Measurable™

CWE – CWE Field Completeness Goals

http://cwedev1.mitre.org/data/reports/field_completeness_goals.html

Common Weakness Enumeration

A Community-Developed Dictionary of Software Weakness Types

Home > CWE List > Reports > CWE Field Completeness Goals | View the CWE List

CWE List
Full Dictionary View
Classification Tree
Leaf Nodes
Other Views
Reports
About
Sources
Process
Documents
Community
Related Activities
Discussion List
Research
News
Calendar
Free Newsletter
Compatibility
Program
Requirements
Declarations
Make a Declaration
Contact Us
Search the Site

CWE Field Completeness Goals

Search by ID

Section Contents
CWE List
Full Dictionary View
Classification Tree
Leaf Nodes
Other Views
Reports
Other Items of Interest
Sources

Following is a report of each CWE field that identifies its priority, as well as how often it must be completed, from the perspective of CWE content maintenance. The CWE community is encouraged to provide feedback on these priorities.

Key

Completeness	How many nodes and which roles (i.e., Categories, Groups, Types, Variants or All) should have this field filled in. This will be the basis of measuring one type of progress towards the first official version of CWE. • **All**: All nodes need the field. • **Most [role]**: Most nodes with the given role are likely to need the field. • **Some [role]**: Some nodes with the given role use the field; an empty value is typical.
Priority	What the CWE team thinks the priority should be, as of January 2008 • **1**: Essential for defining what a node is about • **2**: Important for making CWE useful to most stakeholders • **3**: Important for informational or educational purposes, but not critical to the definition of the node itself; or, important for an influential stakeholder, but not most. • **4**: Useful for informational or educational purposes • **5**: Very few known use-cases; outside the scope of CWE itself; or not yet well-defined
Schema	What schema changes might be required in the future
Content	Considerations for content of this field
Community	Opportunities for community involvement

Field Completeness Requirements Summary

Affected_Resource	**Completeness:**	some Types, some Variants
	Priority:	4
	Schema:	possibly small changes, depending on view formalization
	Community:	could help define additional resources and populate nodes
Alternate_Terms	**Completeness:**	some Types, some Variants, some Categories
	Priority:	4
Applicable_Platforms	**Completeness:**	all Types, all Variants, all Categories
	Priority:	2
	Schema:	need to handle "All affected, but of greatest concern in language X"; also need to handle environment/OS

Making Security Measurable™

Guiding CWE's Changes By Stakeholder Priorities

Stakeholder Field Priorities

Identified Stakeholders | Stakeholder Field Priorities

This report lists the active CWE fields and attempts to characterize their relative importance to various stakeholders. Because there are many different uses for CWE, this report will help the CWE Content Team to prioritize content management activities.

Identified Stakeholders

Type	Tier	Description
Devel	2	Developers, designers, architects, and vendors of software, whether it is commercial or open source, customized or widely available. Could also be Assessment Customers. Note: this group includes the internal development team, any contracted third-party developers, and the marketing/support teams who act as the interface to customers.
Scan Vend	1	Assessment Vendors - Developers of code scanners, services, and other types of assessment technologies.
Educ	2	Educators or certification programs that teach developers how to develop more secure code, and/or how to find vulnerabilities.
IEC	2	ISO/IEC Project 22.24772, which is developing "Guidance for Avoiding Vulnerabilities through Language Selection and Use"
SAMATE	1	The Software Assurance Metrics and Tool Evaluation (SAMATE) project
Formal	2	The CWE Formalization project, led by KDM Analytics.
CWE	1	The CWE content team itself, both for maintenance and longer-term goals.
Cust	2	Customers who buy software.

BACK TO TOP

Stakeholder Field Priorities

Key

- **Req** The field is essential for the stakeholder (rating: 20 for Tier 1, 10 for Tier 2)
- **Imp** The field is important for the stakeholder (rating: 10 for Tier 1, 5 for Tier 2)
- **Nice** The field is convenient for the stakeholder, but its absence will not hamper operations (rating: 2 for Tier 1, 1 for Tier 2)
- **-** Not needed by the stakeholder (rating: 0)

Field	Score	Devel	Scan Vend	Educ	IEC	SAMATE	Formal	CWE	Cust
Name	110	**Req**	**Req**	**Req**	**Req**	**Req**	**Req**	**Req**	**Req**
Description	110	**Req**	**Req**	**Req**	**Req**	**Req**	**Req**	**Req**	**Req**
White_Box_Definition	62	Nice	**Req**	Nice	-	**Req**	**Req**	Imp	-
Node_Relationship	62	**Req**	Imp	**Req**	Nice	Imp	Nice	**Req**	-
Data: Chains and Composites	58	Nice	Imp	Nice	Imp	**Req**	Nice	**Req**	-
Status	58	Nice	**Req**	Imp	Nice	Imp	Nice	**Req**	-
Weakness Abstraction	52	Nice	Imp	Nice	-	**Req**	Imp	Imp	Imp
Time_of_Introduction	46	Imp	Imp	Nice	Imp	Imp	Imp	Imp	-
Data: Natural Hierarchy View	44	Imp	Imp	Imp	Nice	Nice	Nice	**Req**	-
Applicable_Platforms	44	**Req**	Imp	Imp	**Req**	Nice	-	Nice	Imp
Related_Attack_Patterns	41	**Req**	Imp	Imp	Nice	-	-	Imp	Imp
Detection_Factor	37	Nice	Imp	-	-	**Req**	Imp	-	Nice
Observed_Examples	34	Imp	Imp	Imp	Nice	Nice	Nice	Imp	-
Causal_Nature	34	Imp	Imp	Nice	Nice	Imp	Imp	Nice	-
Demonstrative_Example	34	Imp	Imp	Imp	Nice	Nice	Nice	Imp	-
Likelihood_of_Exploit	33	Imp	Imp	Imp	Nice	Nice	-	-	**Req**
Common_Consequences	32	**Req**	Nice	**Req**	-	-	-	-	**Req**
Enabling_Factors_for_Exploitation	28	Imp	Imp	Nice	Nice	Imp	Nice	-	-
Potential_Mitigations	28	**Req**	Nice	**Req**	Nice	Nice	Nice	Nice	-
Alternate_Terms	27	Imp	Imp	Nice	-	-	-	Imp	Nice
Context_Notes	26	Nice	Nice	Nice	-	Nice	-	**Req**	-
Source_Taxonomy	25	-	Nice	Nice	-	Nice	-	**Req**	-
Taxonomy_Mapping	25	Nice	Imp	Nice	-	Nice	-	Imp	Nice
Weakness_Ordinality	25	Nice	Nice	Nice	-	Imp	Nice	Imp	-
Research_Gaps	21	-	Nice	Nice	-	Nice	Imp	Imp	Nice
Relevant_Properties	20	Nice	-	Imp	Nice	Nice	Nice	Imp	-
References	20	Imp	-	Imp	-	-	-	Imp	-
Functional_Area	19	Imp	Nice	Imp	Imp	-	-	Nice	-
Affected_Resource	16	**Req**	Nice	Nice	Nice	-	Nice	-	Nice
Black_Box_Definition	12	Imp	Nice	Nice	-	Nice	-	Nice	-

Key: Items in BOLD and GRAY increased over the previous draft. Items that are BOLD without GRAY were at the maximum for a previous draft, but only increased because more nodes were added. Items marked Up or Down had at least a 3% change.

Pri	Field	v5	v7	v9
200	CWE_ID	599	**634**	**695**
110	Name	599	**634**	**695**
110	Description	521	**634**	**695**
62	Node_Relationship	599	**634**	**687**
62	Type	599	**634**	**695**
62	White_Box_Definition			**12**
46	Time_of_Introduction	1	1	**82**
44	Applicable_Platforms	468	**486**	**522**
41	Related_Attack_Patterns			**154**
37	Detection_Factor			**5**
34	CVEs_Mentioned	203	**223**	**242**
34	Observed_Example	205	**225**	**252**
34	Demonstrative_Example	163	**173**	**184**
34	Causal_Nature	66	**67**	**75**
33	Likelihood_of_Exploit	115	115	**130**
32	Common_Consequences	105	**106**	**130**
28	Potential_Mitigations	328	**345**	**379**
28	Enabling_Factors_for_Exploitation		**1**	**15**
27	Alternate_Terms	24	**30**	**38**
26	Context_Notes	339	**360**	**382**
25	Source_Taxonomy	529	**541**	**542**
25	Weakness_Ordinality	68	**82**	**97**
21	Research_Gaps	52	**57**	**61**
20	Relevant_Properties			**12**
20	References	39	**50**	**61**
19	Functional_Area	28	28	28
16	Affected_Resource		**50**	**52**
12	Black_Box_Definition			
1	Common_Methods_of_Exploitation	2	2	

Field	Type	Variant	Category	Grouping
Total	265	256	55	77
CWE_ID	265 (100%)	256 (100%)	55 (100%)	77 (100%)
Name	265 (100%)	256 (100%)	55 (100%)	77 (100%)
Type	265 (100%)	256 (100%)	55 (100%)	77 (100%)
Description	265 (100%)	256 (100%)	55 (100%)	77 (100%)
Node_Relationship	265 (100%)	256 (100%)	55 (100%)	77 (100%)
References	30 (11%)	18 (7%)	8 (14%)	4 (5%)
Applicable_Platforms	242 (91%)	202 (78%)	39 (70%)	27 (35%)
Context_Notes	191 (72%)	135 (52%)	24 (43%)	14 (18%)
Research_Gaps	30 (11%)	17 (6%)	6 (10%)	6 (7%)
Demonstrative_Example	100 (37%)	71 (27%)	7 (12%)	3 (3%)
Observed_Example	127 (47%)	106 (41%)	11 (20%)	1 (1%)
Weakness_Ordinality	43 (16%)	37 (14%)	9 (16%)	(0%)
Alternate_Terms	22 (8%)	8 (3%)	4 (7%)	(0%)
Functional_Area	17 (6%)	3 (1%)	3 (5%)	5 (6%)
Likelihood_of_Exploit	79 (29%)	42 (16%)	6 (10%)	1 (1%)
Causal_Nature	35 (13%)	30 (11%)	9 (16%)	(0%)
Affected_Resource	22 (8%)	18 (7%)	5 (9%)	5 (6%)
Source_Taxonomy	238 (89%)	218 (85%)	43 (78%)	42 (54%)
CVEs_Mentioned	121 (45%)	102 (39%)	11 (20%)	1 (1%)
Common_Methods_of_Exploitation	(0%)	(0%)	(0%)	(0%)
Enabling_Factors_for_Exploitation	8 (3%)	7 (2%)	(0%)	(0%)
Time_of_Introduction	7 (2%)	7 (2%)	(0%)	1 (1%)
Potential_Mitigations	159 (60%)	184 (71%)	25 (45%)	10 (12%)
Common_Consequences	80 (30%)	41 (16%)	5 (9%)	2 (2%)
Related_Attack_Patterns	73 (27%)	44 (17%)	26 (47%)	11 (14%)
White_Box_Definition	8 (3%)	4 (1%)	(0%)	(0%)
Black_Box_Definition	(0%)	(0%)	(0%)	(0%)

CWE – CWE Usage Scenarios

http://cwe.mitre.org/community/research/usage_scenarios.html

Common Weakness Enumeration

A Community-Developed Dictionary of Software Weakness Types

Home > Community > Research > CWE Usage Scenarios | View the CWE List

CWE List: Full Dictionary View, Classification Tree, Leaf Nodes, Other Views
About: Sources, Process, Documents
Community: Related Activities, Discussion List, Research
News: Calendar, Free Newsletter
Compatibility: Program, Requirements, Declarations, Make a Declaration
Contact Us: Search the Site

CWE Usage Scenarios

CWE Usage Scenarios | Stakeholder Analysis | Views

Section Contents: Community, Related Activities, Discussion List, CWE Research, Discussion List Archives, Working Documents

Usage Modes

- **Browse**: navigate or browse through the CWE, following related nodes or finding knowledge gaps
- **Search**: search for specific CWE IDs
- **Lookup**: look up a particular CWE whose ID is known
- **Inspect**: Learn additional details about a particular CWE

Usage Scenarios

Mapping	The user has a specific weakness/attack/vulnerability in mind and needs to find the CWE identifier for it. Modes: Browse, Search Considerations: abstraction differences may be a factor during mapping. Need to match expectations of the mapper and support alternate terminology.
Compare	The user needs to compare multiple tools or repositories in terms of their coverage and focus. Or, the user wants to compare multiple applications in terms of their "weakness density." Modes: Lookup, Inspect, Search
Learn More	The user needs to learn more about a specific issue. Modes: Lookup, Inspect, Search, Browse
Find Gaps	The user wants to learn about new CWEs that might not be covered (by the user's knowledge, a tool, etc.) Modes: Browse, Search
Find Related	The user is working from a specific CWE and wants to learn about related CWEs. Modes: Browse, Search
Prioritize	The user needs to find the highest-priority entries, for some definition of "priority". Modes: Search, Lookup, Inspect
Announce a Vulnerability	The user wants to publicly announce a vulnerability and use a CWE ID in the announcement. Modes: Browse, Search Considerations: abstraction differences may be a factor during mapping. Need to

Making Security Measurable™

CWE – CWE Views

http://cwe.mitre.org/community/research/views.html

Common Weakness En… A Community-Developed Dictionary of S…

Home > Community > Research > CWE Views

CWE Views

CWE Usage Scenarios | Stakeh…

User Issues/Questions

- Is this the right view? What are the others?
- Is the navigation/structure natural? Does it allow…

Types of Views

- Lists: simple lists of CWE nodes for a specialized…
- Organization Schemes: hierarchical or other orga… purpose.

Views

V1	Programming language-specific When programming or analyzing specific la… issues of which you should be aware. Also, … related characteristics.
V2	Platform-specific When a program is run on a platform (Win… (32/64 bit, multi-processor), there are cert… addition to the actual language used. E.g., … concurrency
V3	Technology-specific Is the weakness generic, or is it primarily a… technology class: Web, OS, Database?
V4	Common Weakness Chains When viewing a weakness, it is useful to kn… in the same place where the result is seen, … or result from a weakness is useful to supp… weakness relationships.
V5	Taxonomy/Classification From a more formal taxonomic perspective, … various weaknesses may be important.
V6	Commonality How easy is it for someone to make this mi…

CWE – CWE List (Draft 9)

http://cwe.mitre.org/data/index.html

Common Weakness Enumeration

A Community-Developed Dictionary of Software Weakness Types

Home > CWE List (Draft 9) | View the CWE List

CWE List (Draft 9)

Comprehensive Views | Graphs | Explicit Slices | Implicit Slices

The Common Weakness Enumeration (CWE™), currently in a very preliminary form, is a list of software weaknesses. Creating the list is a community initiative. Together, these organizations and any others that wish to join the effort, are creating specific and succinct definitions for each of the elements in the CWE List. By leveraging the widest possible group of interests and talents we hope to ensure that the CWE elements are adequately described and differentiated. The next steps are to adequately capture the specific effects, behaviors, exploit mechanisms, and implementation details in the CWE dictionary as well as to review and revise the presentation approaches that will best suit this information.

Comprehensive Views

CWE XML Draft 9 (2020 KB)	XSD schema (59 KB)
CWE Comprehensive Dictionary	Natural Hierarchy
1-Page Graphical PDF	(149 KB)

Graphs

(1000)	Natural Hierarchy	Graph
(631)	Resource-specific Weaknesses	Graph

Explicit Slices

(604)	Deprecated	List	Slice	XML
(629)	Weaknesses in OWASP Top Ten	List	Slice	XML
(630)	Weaknesses Examined by SAMATE	List	Slice	XML
(635)	Weaknesses Used by NVD	List	Slice	XML

Implicit Slices

(2000)	Comprehensive CWE Dictionary	List	Slice	XML
(658)	Weaknesses found in the C Language	List	Slice	XML
(659)	Weaknesses found in the C++ Language	List	Slice	XML
(660)	Weaknesses found in the Java Language	List	Slice	XML
(661)	Weaknesses found in the PHP Language	List	Slice	XML
(677)	Weakness Base Elements	List	Slice	XML
(678)	Composites	List	Slice	XML
(679)	Chain Elements	List	Slice	XML

Please contact cwe@mitre.org with suggestions for additional views.

Page Last Updated: April 11, 2008

Homeland Security

CWE is sponsored by the National Cyber Security Division of the U.S. Department of Homeland Security.
This Web site is hosted by The MITRE Corporation.
Copyright 2008, The MITRE Corporation. CWE and the CWE logo are trademarks of The MITRE Corporation.
Contact cwe@mitre.org for more information.

Privacy policy | Terms of use | Contact us

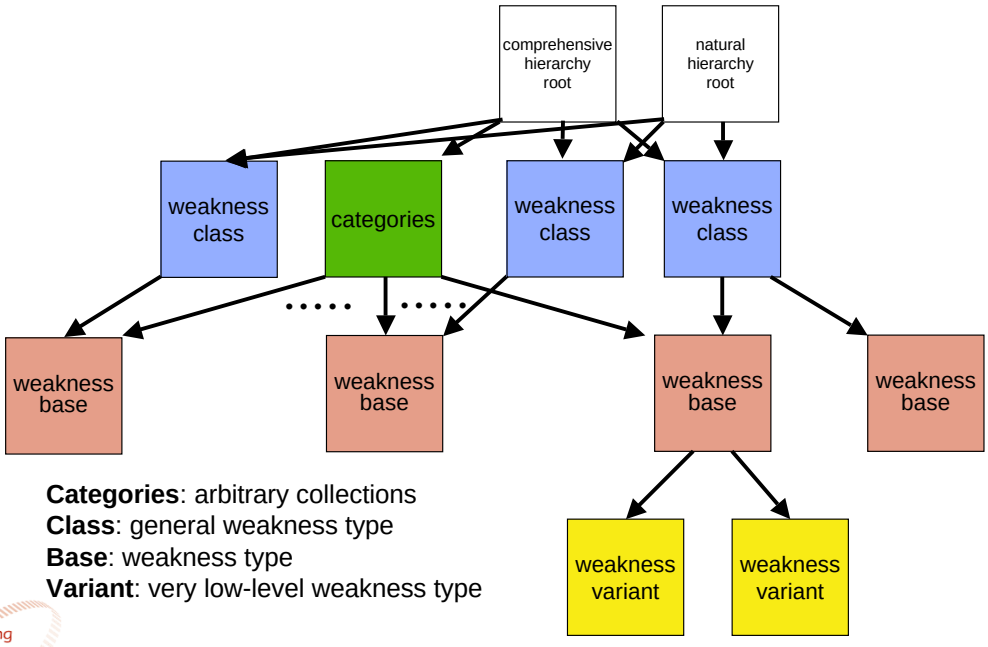
Making Security Measurable™

CWE-1000: Natural Hierarchy

- Pillars, Classes, Bases, Variants
- Why a hierarchy?
  - **Simplify CWE mapping**
  - **Identify gaps**
    - Tool coverage
    - General knowledge
  - **Education**
- Challenges
  - **Everything's primary and resultant**
  - **Chains are everywhere**
  - **Can be difficult to remove "attack" from the "weakness"**
  - **Terminology is vague or has multiple uses**
    - "Leak" has at least 2 meanings
  - **Many well-known issues are really consequences of errors that aren't so well-known**
    - SQL injection, buffer overflow, NULL pointer dereference

Making Security Measurable™

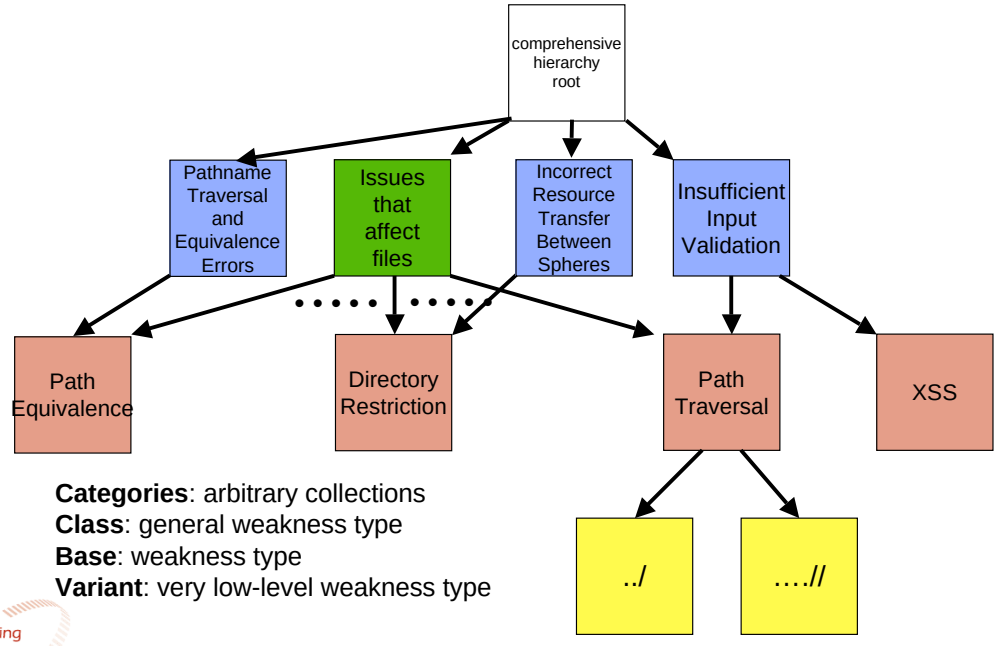
Node Types



Categories: arbitrary collections
Class: general weakness type
Base: weakness type
Variant: very low-level weakness type

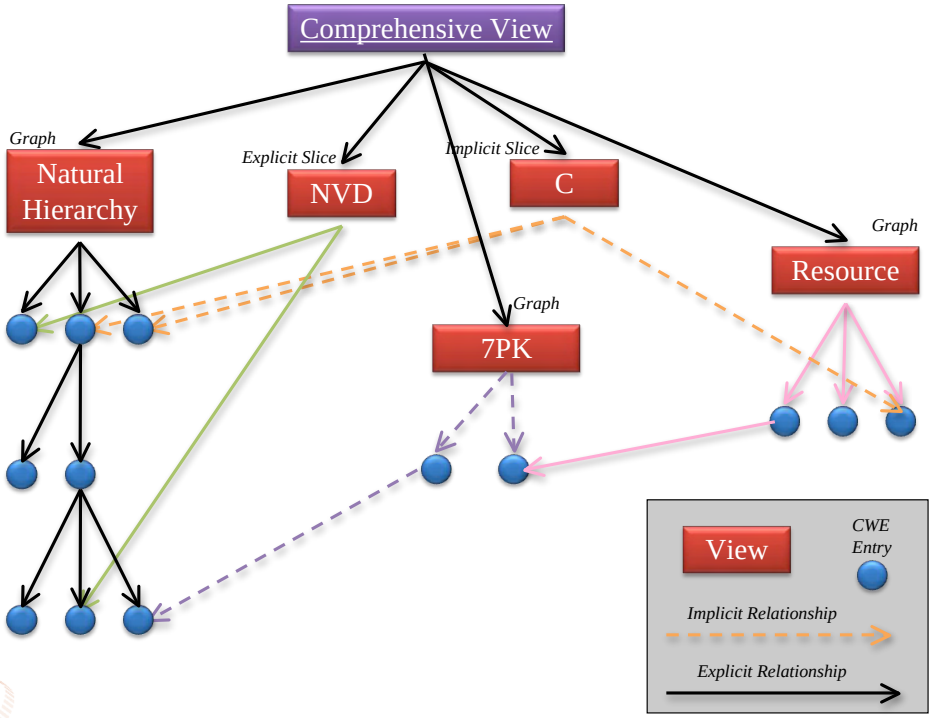
Making Security Measurable™

Node Types: Example



Categories: arbitrary collections
Class: general weakness type
Base: weakness type
Variant: very low-level weakness type

Making Security Measurable™



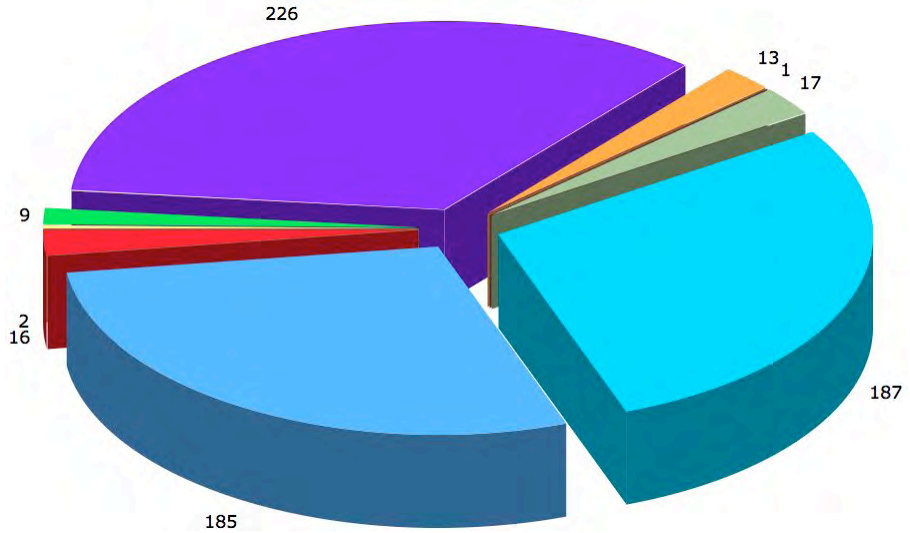
Making Security Measurable™

Problems With Categorization

- Same terminology used in multiple dimensions
  - **Frequent mix of attacks, threats, weaknesses/faults, consequences**
    - e.g. buffer overflows, directory traversal
  - **Hard to classify vulnerabilities cleanly**
  - **Complex issues don't have simple terms**
- Lack of diagnosis by researchers
  - **Primary vs. resultant issues**
- Few disclosures offer enough details
  - **CWE has almost 700 entries**
  - **Many variants of buffer overflow, XSS, directory traversal**
- UI implications in data collection
  - **How to consistently select from 700 options?**
  - **How to go back and re-categorize when you identify a new type of issue?**

Making Security Measurable™

Role Analysis: Classes, Bases, Variants, and Categories

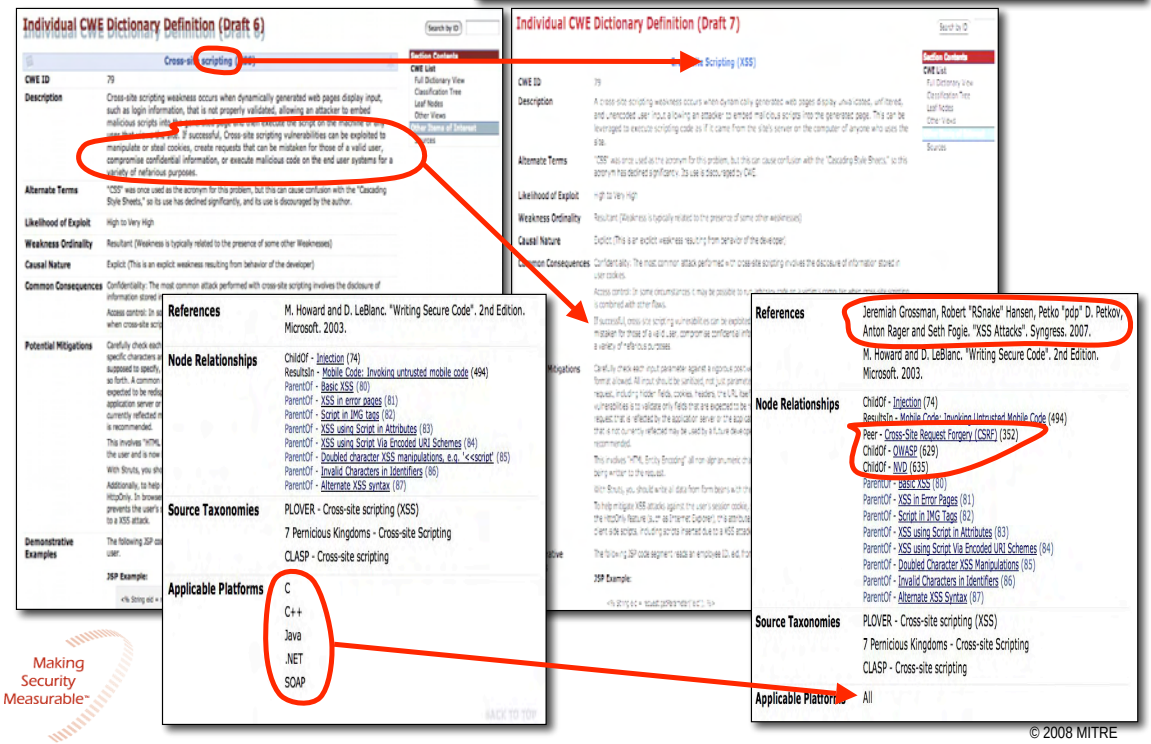


185	Variants without Base/Variant as a Parent
16	Bases with a Parent that's not Class/Category
2	Bases with a child that is a Class/Category
9	Bases with children, where non are a Variant
226	Bases without any children
13	Classes without any children
1	Classes with a Parent that is not a Class/Category
17	Classes with a Child that is a Variant
187	Nodes with no Parent/Child issues

Making Security Measurable™

CWE-79 Cross-site scripting (XSS) (Draft 6to7)

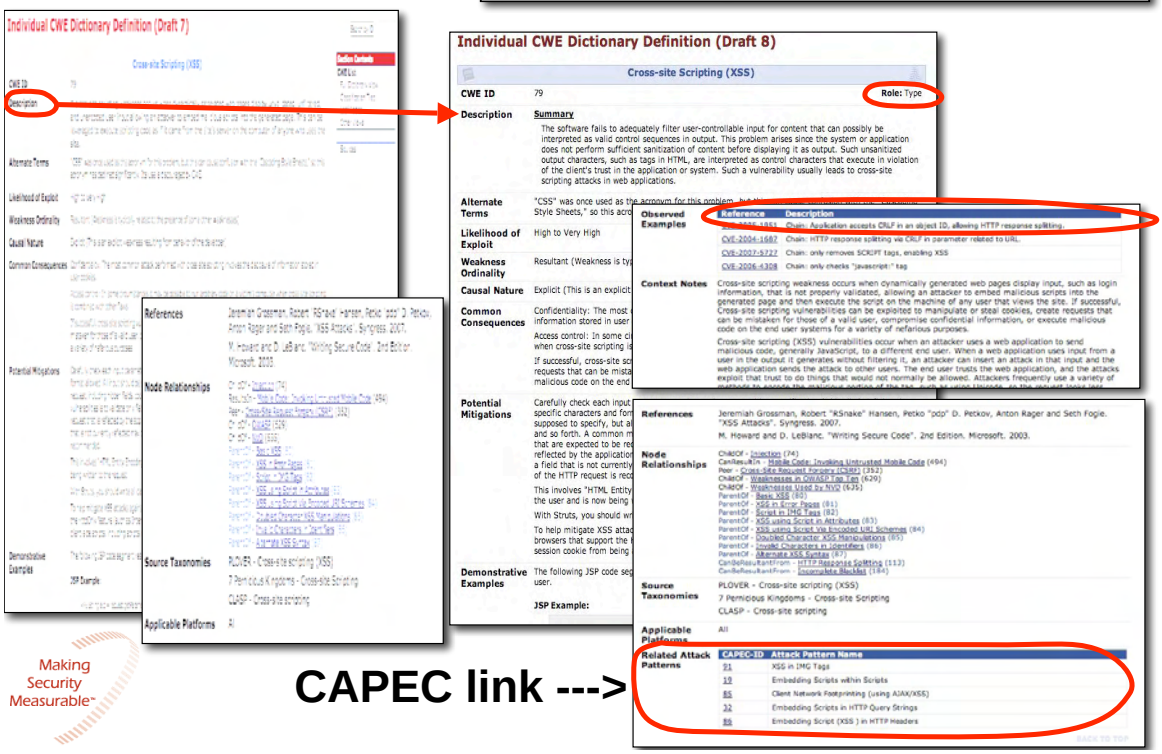
[cwe.mitre.org/data/definition/79.html]



Making Security Measurable™

CWE-79 Cross-site scripting (XSS) (Draft 7to8)

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

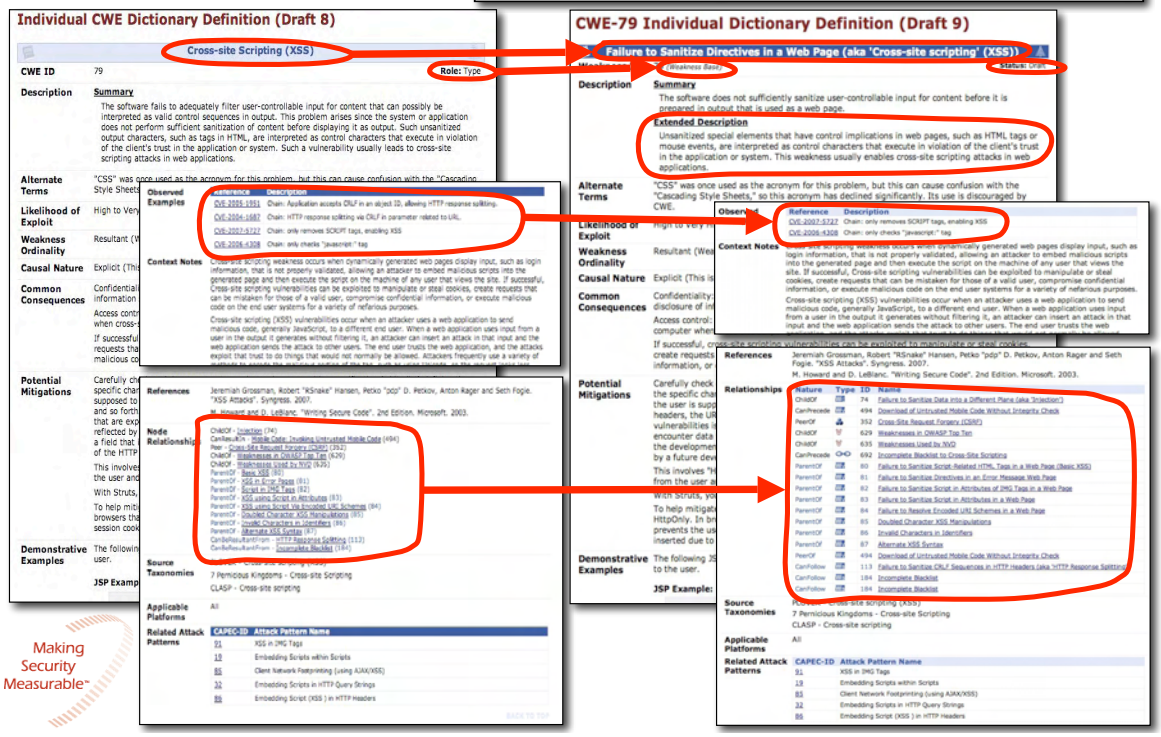


Making Security Measurable™

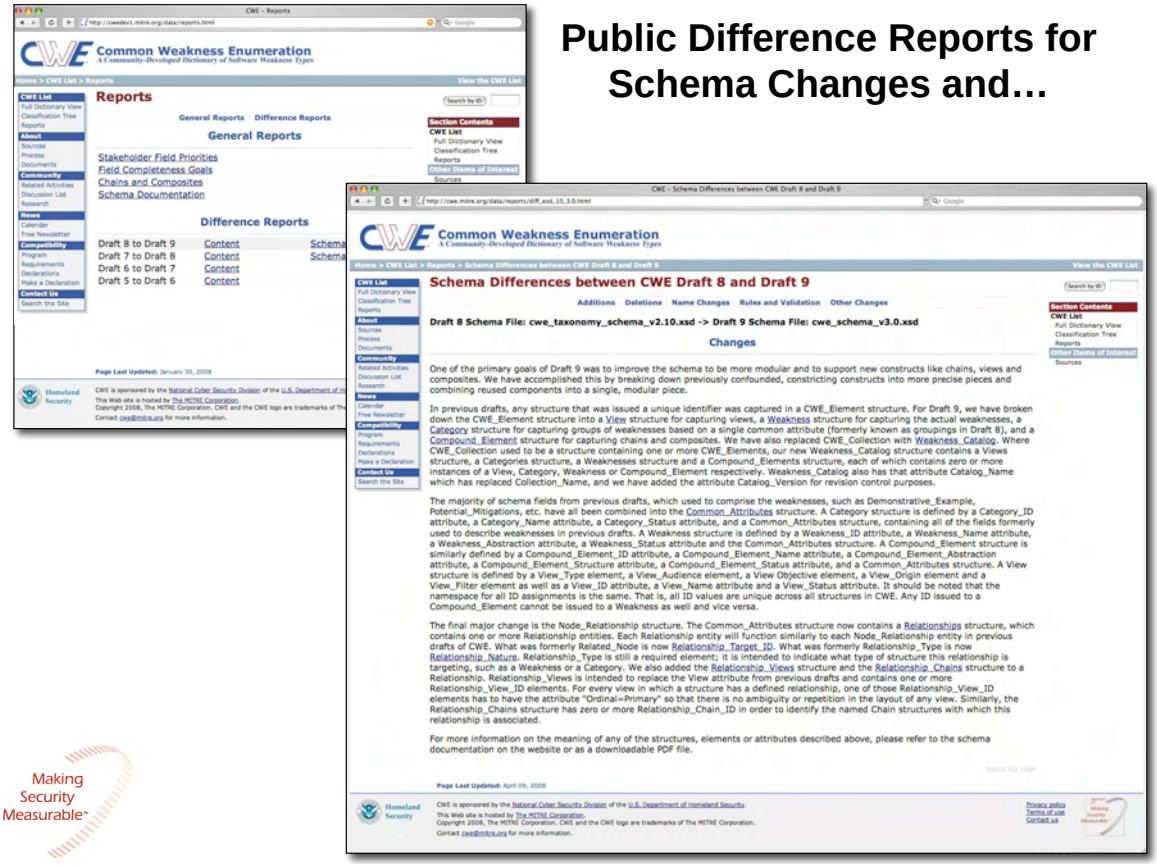
CAPEC link --->

CWE-79 Failure to Sanitize Directives in a Web Page (aka 'Cross-site scripting' (XSS)) (Draft 8to9)

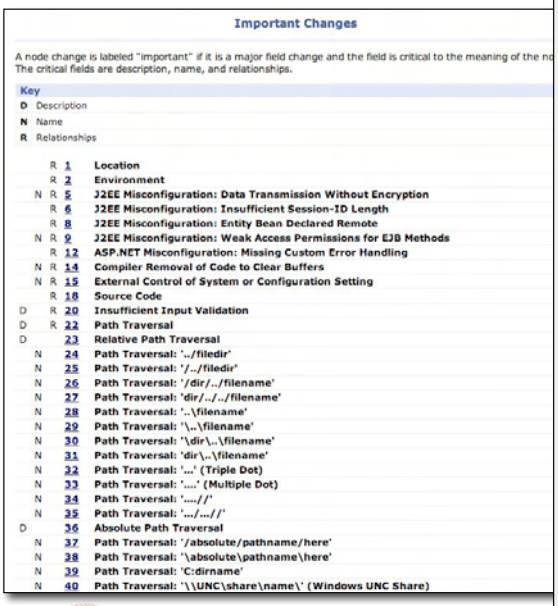
[cwe.mitre.org/data/definition/79.html]

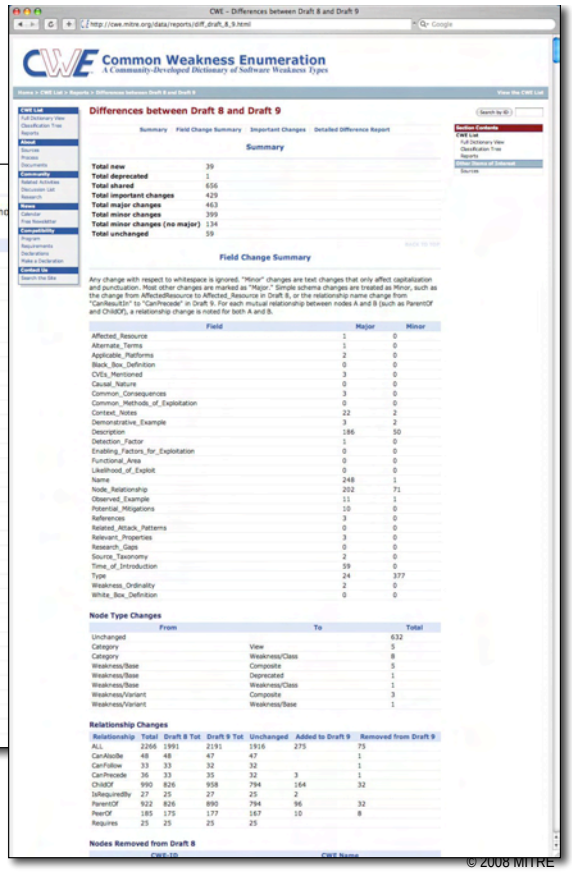


Public Difference Reports for Schema Changes and…



Public Difference Reports for Content Changes

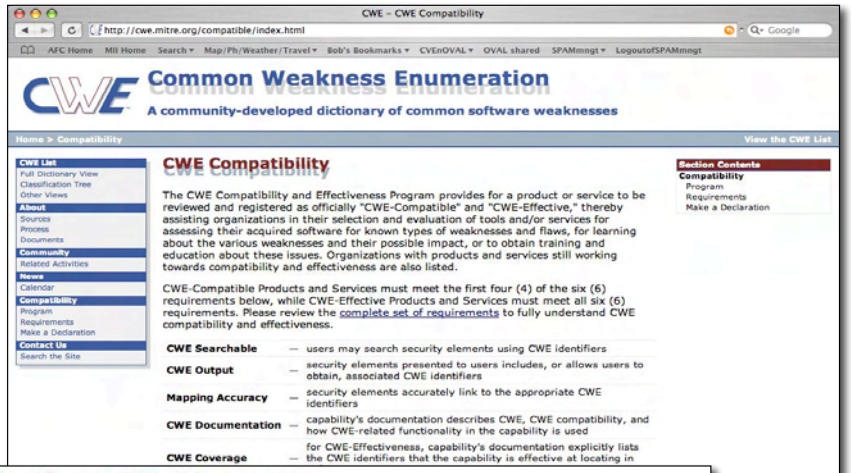


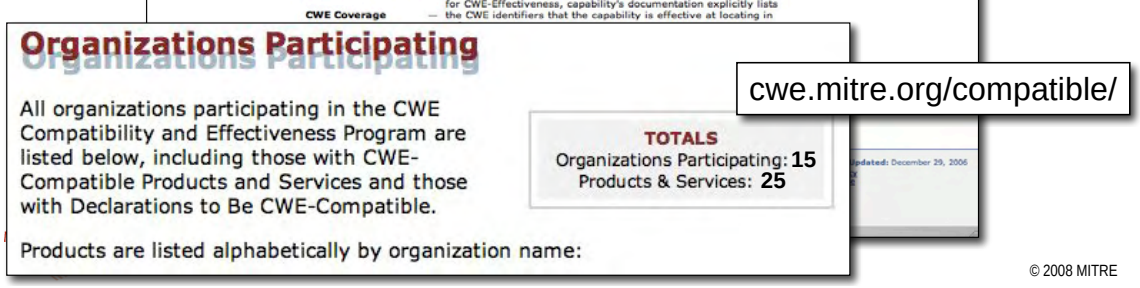


Making Security Measurable™

CWE Compatibility & Effectiveness Program

(launched Feb 2007)



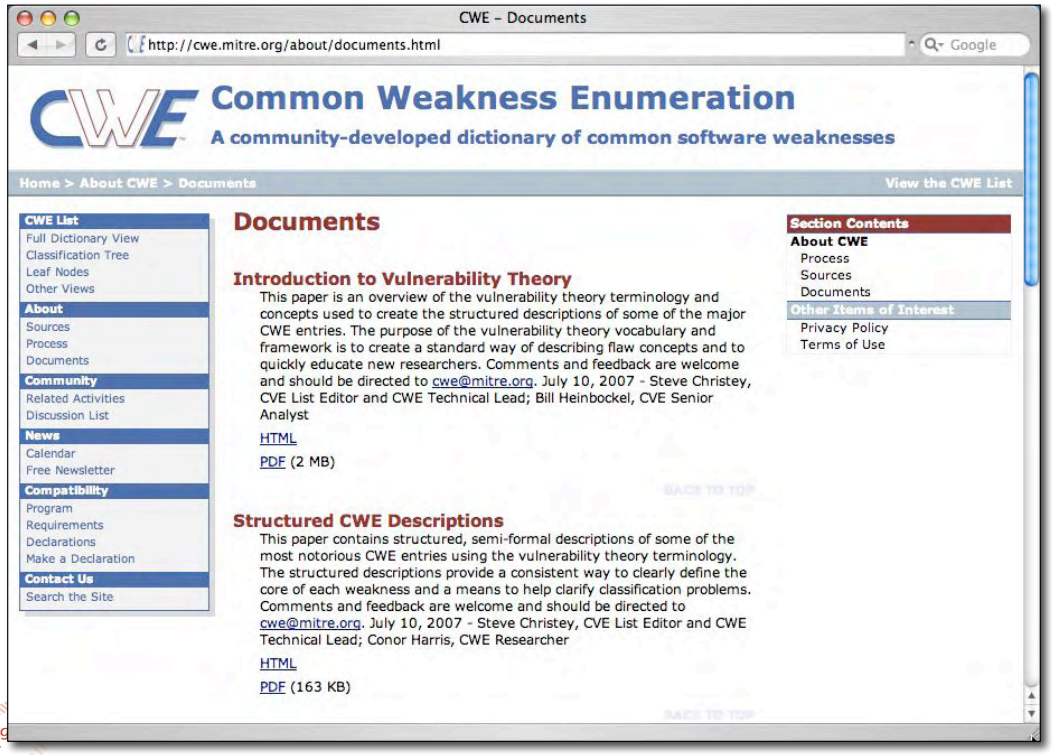


Current Community Contributing to the Common Weakness Enumeration

- AppSIC
- Apple
- Aspect Security
- Booz Allen Hamilton Inc.
- Cenzic
- CERIAS/Purdue University
- CERT/CC
- Cigital
- Codenomicon
- Core Security
- Coverity
- DHS
- Fortify
- Gramma Tech
- IBM
- Interoperability Clearing House
- JHU/APL
- JMU
- Kestrel Technology
- KDM Analytics
- Klocwork
- McAfee
- Microsoft
- MIT Lincoln Labs
- MITRE
- North Carolina State University
- NIST
- NSA
- OMG
- Oracle
- Ounce Labs
- OSD
- OWASP
- Palamida
- Parasoft
- PolySpace Technologies
- proServices Corporation
- SANS Institute
- SecurityInnovation
- Security University
- Semantic Designs
- SofCheck
- SPI Dynamics
- SureLogic, Inc.
- Symantec
- UNISYS
- VERACODE
- Watchfire
- WASC
- Whitehat Security, Inc.

Making Security Measurable™

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



Making Security Measurable™

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**
- 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?"**

Making Security Measurable™

Vulnerability Theory

- A framework for systematically understanding and discussing vulnerabilities and related concepts
- I am not going to define "vulnerability" here. You're welcome.
- Common Weakness Enumeration (CWE) lists almost 700 entries!
  - **Buffer overflow, off-by-one, XSS, format string, unprotected communications channel, weak permissions, incorrect blacklist, use of hard-coded crypto key, insufficient randomness, …**
  - **Weaknesses can lead to vulnerabilities**
- Terminology is sorely lacking

With so many issues, we cannot reasonably expect developers (or tools) to anticipate every problem in every line of written code.

Making Security Measurable™

What's it Good For?

- Education
  - **Make the expert's knowledge more accessible to others**
  - **See issues in a new light**
  - **Institutional memory, please!**
- Increase our own understanding of vulnerabilities
  - **Conduct more systematic gap analysis**
  - **Researchers can identify new opportunities**
  - **Richer vocabulary for more precise discussion**
  - **Guidance for vulnerability classification**
  - **Relate issues that seem different on the surface**

Making Security Measurable™

Behaviors, Properties, and Manipulations

A **PRODUCT** implements **FEATURES** by performing certain **BEHAVIORS** that operate on **RESOURCES**. The behaviors can be "atomic" or "f
used at the tir
to a different
socket, memo
cookies, heade

The product's
by preserving
encoding man
handled as AS
"equivalent" to
if the meaning

The developer
appropriate be
an **IMPLEMEN**
intended polic
policies are th
the product.

Types of Manipulations

There are three main manipulation types:

REACHABILITY - required to reach the relevant behavior. Example - when a buffer overflow can only occur in the password field, the reachabi

TRIGGE
request

FACILI
limitatio
alphanu
Perl/PHP
arbitrar
manipul
already

Manipul
compos

Artifact Labels

Artifact Labels are used to mark important locations in code, design, or algorithm that are relevant to a vulnerability. Vulnerability researchers frequently highlight these locations when presenting vulnerable code, but they don't use the same terminology, if at all. These labels turn out to be useful in describing certain vulnerability "topology" in the abstract sense, as well.

NOTE: recommendations for alternate terms are welcome.

INTERACTION POINT - a location in the product where "input" (of either data or directives) enters the system. "Injection" might be a more natural term, but it's already overloaded, and it seems to be data-centric. This is equivalent to what others call "entry points," but that term has different uses in binary reverse engineering.

CROSSOVER POINT - the location in which an expected property is violated. Any subsequent behaviors that depend on the expected property could be subject to a vulnerable condition. A crossover point could occur *between* lines of code, e.g. if a product-generated filename is never checked for directory traversal sequences.

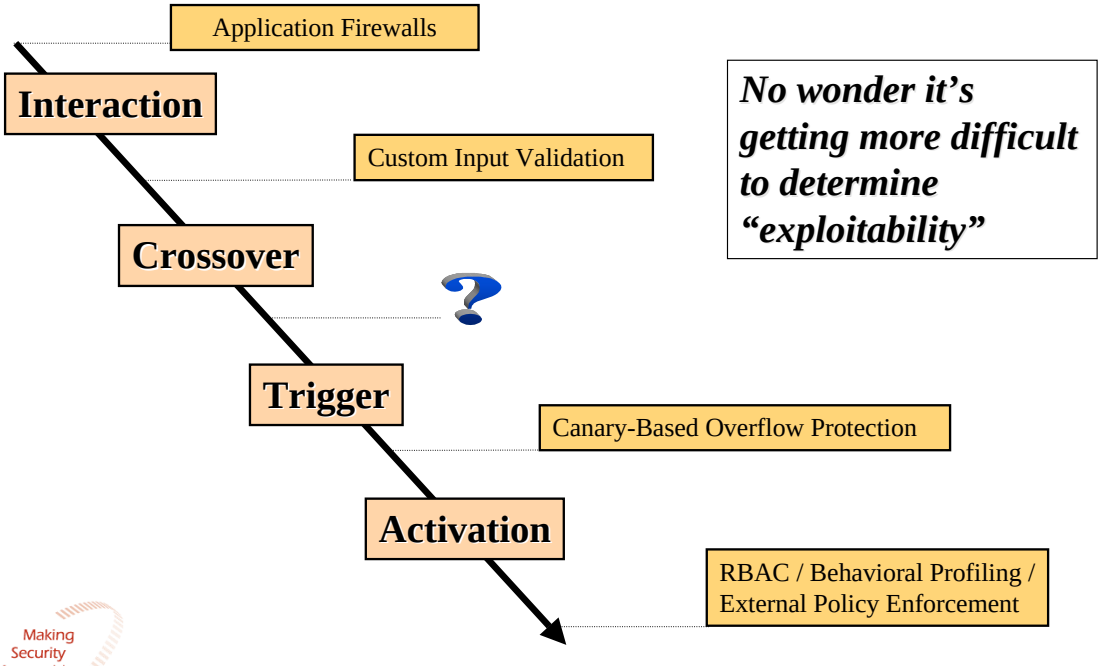
TRIGGER POINT - the location in the product where a "fault" occurs, and the product can no longer stop itself from performing incorrect behaviors in the future.

ACTIVATION POINT - the location where the attacker's "payload" becomes activated; presumably the payload involves the incorrect behaviors.

ATTACK VECTOR - a tuple of (PRIMARY FAULT, RESULTANT FAULTS, INTERACTION POINT, CROSSOVER POINT, TRIGGER POINT, ACTIVATION POINT). Different attacks could have different trigger and activation points; for example, a buffer overflow intended for DoS would have a different activation point than one intended for code execution.

Making Security Measurable™

Artifact Labels and Protection Mechanisms



No wonder it's getting more difficult to determine "exploitability"

Making Security Measurable™

XSS:

Interaction: 3, 6
Intermediate Fault: 9
Crossover: between 9 and 10
Trigger: 10
Activation: outside of program (when victim views page)

Traversal:

Interaction: 3
Crossover: between 5 and 6
Trigger: 6
Activation: 7 (or 10, depending on attack)

Overflow:

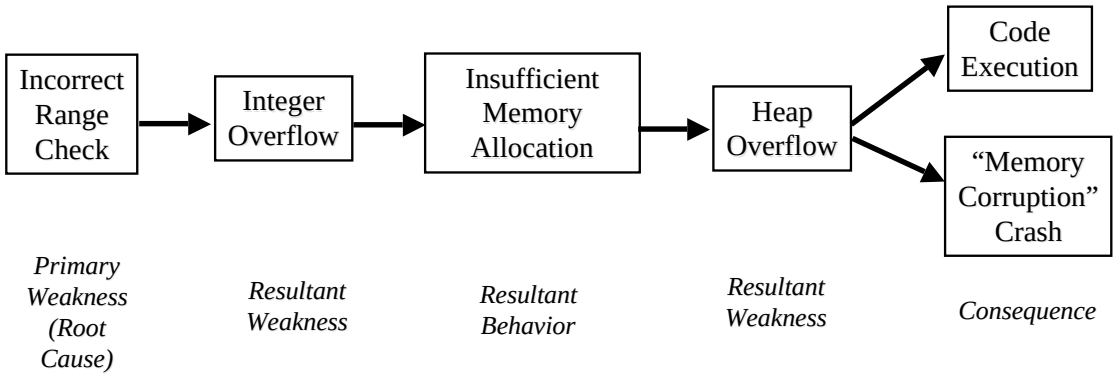
Interaction: 3
Crossover: between 4 and 5
Trigger: 5
Activation: 5 (if DoS intended), outside code (if code execution)

Code 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, ReadMode);
7  ____   while((line=readFile(handle)))
8  ____   {
9  ____      line=stripTags(line, "script");
10 ____      print line;
11 ____      print "<br>\n";
12 ____   }
13 ____   close(handle);
```

Making Security Measurable™

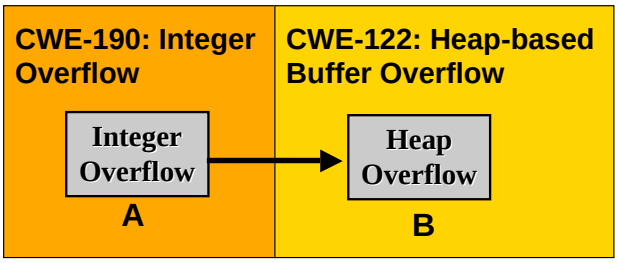
Many Vulnerabilities are Multi-Factor



One of the main problems with classification and terminology is that ANY behavior in the chain could be regarded as the vulnerability.

Making Security Measurable™

Chain Example: Integer Overflow to Heap Overflow



Assumption: height and width are reasonable sizes.

height = 65534; width = 65534

```
A  size = height * width;
   buf = malloc(size);
B  memmove(buf, InputBuf, SZ);
```

The buffer overflow occurs because the newly created buffer is smaller than expected, because the integer overflow causes the 'size' variable to be smaller than expected.

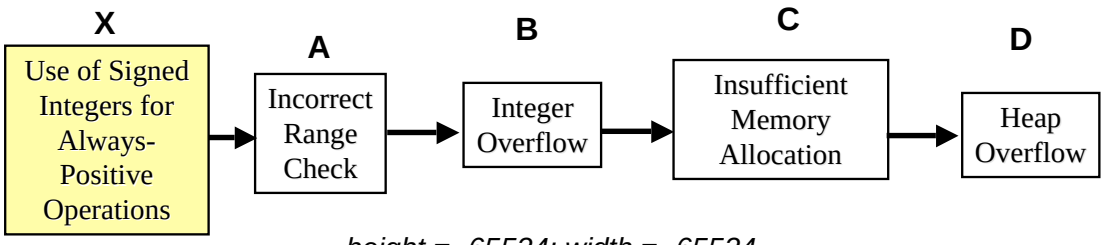
Making Security Measurable™

Protection Schemes

- Definition: a behavior that is intended to protect the product against one or more attacks
- Note: protection schemes (e.g. input validation) are not the same as security features (e.g. encryption, authentication)
- Explicit
  - **"Validate/sanitize input"**
    - This is the vaguest, most-abused phrase today!
  - **"Filter bad characters"**
  - **"Check input file type"**
  - **"Verify number within range"**
  - **"Apply regular expression"**
  - **"Convert to correct data type"**
- Implicit
  - **Data execution prevention, sandboxing, address reordering, taint checking, …**
- Implicit at one level is explicit at another
  - **htmlentities() in PHP is implicit at the application level, explicit within the interpreter itself**

Making Security Measurable™

Chain Example: Failed Protection Mechanism with Resultant Issues



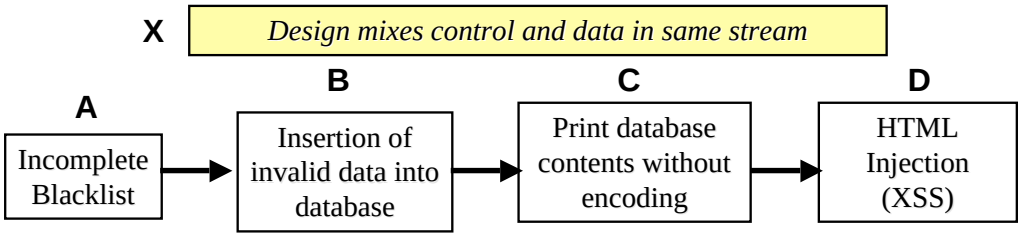
height = -65534; width = -65534

Assumption: the range check will prevent an overflow from occurring.

```
A  if (height > 64000 ||
       width > 64000) {
       error("too big!");
   }
B  size = height * width;
C  buf = malloc(size);
D  memmove(buf, InputBuf, SZ);
```

Making Security Measurable™

Failed Protection Mechanism with Resultant Issues (XSS)



input = <img src="javascript:xyz">

A
B

```
if ($input =~ /<script>/) {
  die("Bad!"); }
AddCommentToDatabase($user, $input);
```

Process 1

D, X
Process 3 (browser)

C

```
while (ReadDb(\$user, \$comment) {
  print "$user:'$comment'<br>"); }
```

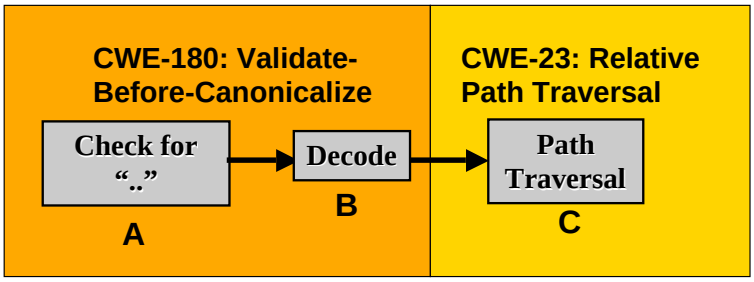
Process 2

Making Security Measurable™

Doing A correctly, implementing encoding in C, or changing X would eliminate this issue.

Chain: Path Traversal with Protection Mechanism Failure

CWE-180 isn't a chain. Neither behavior is an inherent weakness; the only problem is the ordering.



lang = %2e./%2e./%2e/etc/passwd%00

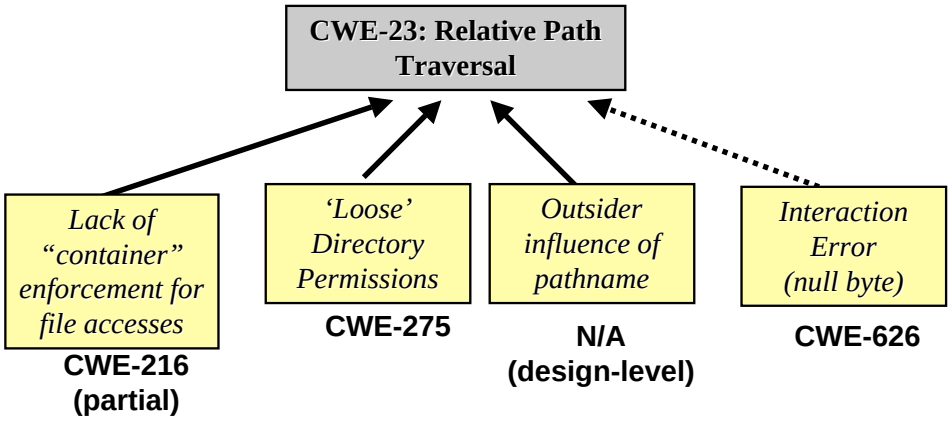
Null byte allows access of files not ending in .txt

A
B
C

```
$home = "/home/www/l18n/";
$input = WebParam('lang');
if ($input =~ /\.\./) {
  die("Bad!"); }
$input = URLdecode($input);
$fname = $home + "$input" + ".txt";
ReadFileAndDumpToUser($fname);
```

Making Security Measurable™

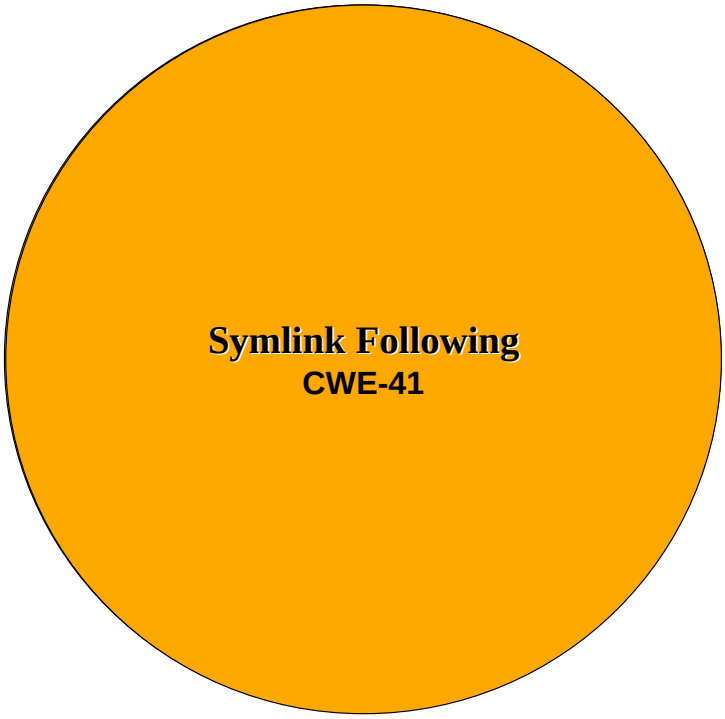
Composite: Path Traversal



- Application can potentially access anywhere user 'nobody' can
- No real built-in OS permissions for 'cannot navigate above this directory'
- Null bytes widen the scope – 'cut off' .txt extension
- Influence of pathname is typically a design-level decision, and can be done safely with proper pathname generation

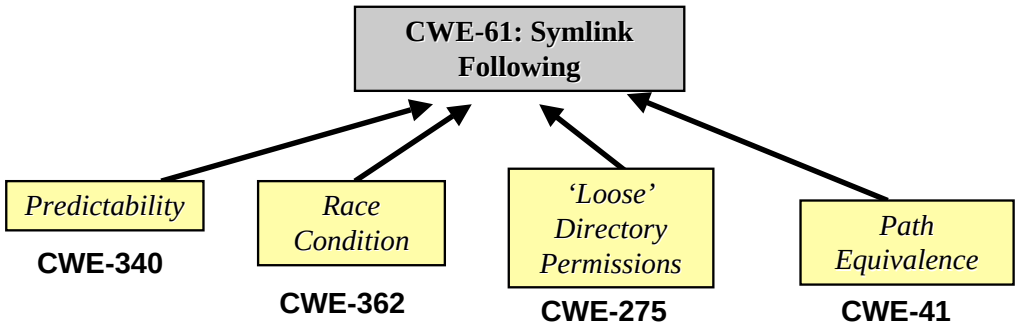
Making Security Measurable™

Symbolic Link Following (composition)



Making Security Measurable™

Composite: Symbolic Link Following



- Filename can be predicted
- File can be created by other party before it is opened for writing
- File created in a shared directory with writable permissions
- Equivalence: a symlink can act an alternate name for a critical file

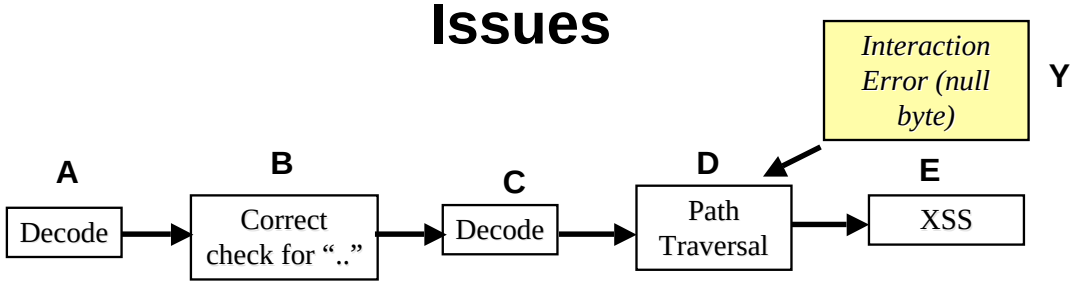
Making Security Measurable™

Integer Overflow Discussion

- Any number of failed protection mechanisms can allow integer overflows
- A canonical integer overflow involves no protection mechanism at all
- Any number of correct protection mechanisms can prevent integer overflows from occurring at all
- Integer overflows can occur, as long as they're checked for (a different type of protection mechanism than the ones that prevent overflows from occurring)
- Integer overflows are expected, valid behavior in some algorithms
- Integer overflows can be relevant in non-memory situations, e.g. loop control

Making Security Measurable™

Incorrect Ordering with Resultant Issues



X

Lack of "container" enforcement for file accesses

lang = %2e./%2e./%2e/logs/apache/weblog%00

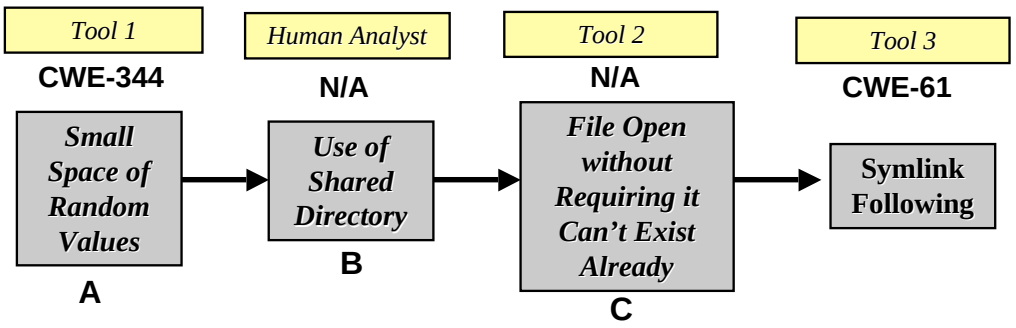
A
B
C
D, E

```
$home = "/home/www/l18n/";
$input = urldecode(WebParam('lang'));
if (($input =~ /^\//) ||
    ($input =~ /\.\./) {
  die("Bad!"); }
$input = urldecode($input);
$fname = $home + "$input" + ".txt";
DumpFileOutput($fname);
```

Null byte allows access of files beyond .txt

Chains, Composites, and Code Scanning

- Comparisons between code scanning capabilities can yield significantly different results
- Very little overlap between tools
  - **… but are they reporting different links in a chain?**



Takeaways from Chains and Composites

- Almost anything can be primary
- Almost anything can be resultant
- Chains are still relatively unexplored
- Resolving a single weakness can break the chain, or reduce the scope
- Behaviors are infinite

Making Security Measurable™

VIEW LIST: CWE-678: Composites (Draft 9)

Composites

View ID	678 *(View)*
Objective	This view (slice) displays only composite weaknesses.
View Data	**Filter Used:** .//@Compound_Element_Structure='Composite'

	CWEs in this view		Total CWEs
Total	9	out of	695
Views	0	out of	14
Categories	0	out of	64
Weaknesses	0	out of	605
Compound_Elements	9	out of	12

- Cross-Site Request Forgery (CSRF) - *(352)*
- Insufficient Control of Filename for Include/Require Statement in PHP
- Permission Race Condition During Resource Copy - *(689)*
- Session Fixation - *(384)*
- Trusting Self-reported IP Address - *(291)*
- Unbounded Transfer ('Classic Buffer Overflow') - *(120)*
- UNIX Symbolic Link (Symlink) Following - *(61)*
- Unrestricted File Upload - *(434)*
- Untrusted Search Path - *(426)*

Composites

CWE-61 UNIX Symbolic Link (Symlink) Following
- CWE-362 Race Condition
- CWE-340 Predictability Problems
- CWE-216 Containment Errors (Container Errors)
- CWE-386 Symbolic Name not Mapping to Correct Object
- CWE-275 Permission Issues

CWE-98 Insufficient Control of Filename for Include/Require Statement in PHP Program (aka 'PHP File Inclusion') *(also a chain link)*
- CWE-456 Missing Initialization *(also a chain link)*
- CWE-473 PHP External Variable Modification *(also a chain link)*
- CWE-425 Direct Request ('Forced Browsing')
- CWE-216 Containment Errors (Container Errors)

CWE-120 Unbounded Transfer ('Classic Buffer Overflow') *(also a chain link)*
- CWE-227 Failure to Fulfill API Contract (aka 'API Abuse')
- CWE-242 Use of Inherently Dangerous Function

CWE-291 Trusting Self-reported IP Address
- CWE-348 Use of Less Trusted Source
- CWE-471 Modification of Assumed-Immutable Data (MAID) *(also a chain link)*

CWE-352 Cross-Site Request Forgery (CSRF)
- CWE-346 Origin Validation Error
- CWE-441 Unintended Proxy/Intermediary
- CWE-642 External Control of User State Data
- CWE-613 Insufficient Session Expiration

CWE-384 Session Fixation
- CWE-346 Origin Validation Error
- CWE-472 External Control of Assumed-Immutable Web Parameter *(also a chain link)*
- CWE-441 Unintended Proxy/Intermediary

CWE-426 Untrusted Search Path
- CWE-216 Containment Errors (Container Errors)
- CWE-275 Permission Issues
- CWE-471 Modification of Assumed-Immutable Data (MAID) *(also a chain link)*

CWE-434 Unrestricted File Upload *(also a chain link)*
- CWE-351 Insufficient Type Distinction
- CWE-436 Interpretation Conflict

CWE-689 Permission Race Condition During Resource Copy
- CWE-362 Race Condition
- CWE-276 Insecure Default Permissions
- CWE-668 Exposure of Resource to Wrong Sphere

Making Security Measurable™

VIEW LIST: CWE-679: Chain Elements (Draft 9)

Chain Elements

View ID 679 (View)

Objective This view (slice) displays only weakness elements that are part of a chain.

View Data **Filter Used:** (./ //Relationship_Nature='CanPrecede') or (@*[contains(name(),'ID')] = //Relationship_Target_ID[../Relationship_Nature='CanPrecede'])

	Count in this view		Total Count
Total	43	out of	695
Views	0	out of	14
Categories	1	out of	64
Weaknesses	36	out of	605
Compound_Elements	6	out of	12

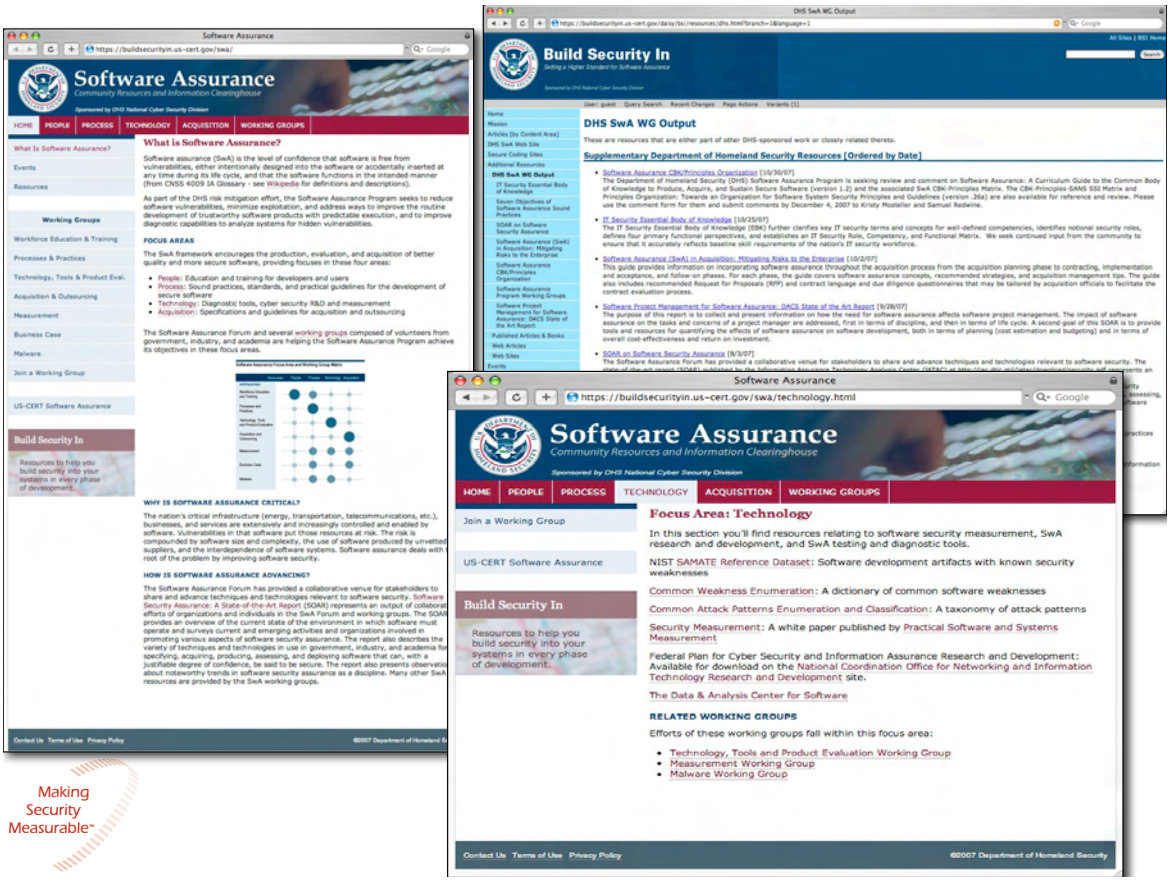
- Authentication Bypass by Alternate Name - (289)
- Cleansing, Canonicalization, and Comparison Errors - (171)
- Design Principle Violation: Client-Side Enforcement of Server-Side Security - (602)
- Design Principle Violation: Reliance on Security through Obscurity - (656)
- Detection of Error Condition Without Action - (390)
- Download of Untrusted Mobile Code Without Integrity Check - (494)
- Error Message Information Leaks - (209)
- External Control of Assumed-Immutable Web Parameter - (472)
- Failure to Catch All Exceptions (Missing Catch Block) - (600)
- Failure to Handle Alternate Encoding - (173)
- Failure to Release Memory Before Removing Last Reference (aka 'Memory Leak') - (401)
- Failure to Resolve Case Sensitivity - (178)
- Failure to Sanitize CRLF Sequences (aka 'CRLF Injection') - (93)
- Failure to Sanitize CRLF Sequences in HTTP Headers (aka 'HTTP Response Splitting') - (113)
- Failure to Sanitize Data into an OS Command (aka 'OS Command Injection') - (78)
- Failure to Sanitize Data into SQL Queries (aka 'SQL Injection') - (89)
- Failure to Sanitize Directives in a Web Page (aka 'Cross-site scripting' (XSS)) - (79)
- Hard-Coded Password - (259)
- Heap-based Buffer Overflow - (122)
- Improper Null Termination - (170)
- Incomplete Blacklist - (184)
- Incomplete Blacklist to Cross-Site Scripting - (692)
- Incorrect Output Sanitization for Logs - (117)
- Insufficient Control of Filename for Include/Require Statement in PHP Program (aka 'PHP File Inclusion') - (
- Integer Overflow (Wrap or Wraparound) - (190)
- Integer Overflow to Buffer Overflow - (680)
- Missing Initialization - (456)
- Modification of Assumed-Immutable Data (MAID) - (471)
- NULL Pointer Dereference - (476)
- Off-by-one Error - (193)
- Path Equivalence: '/multiple/trailing/slash//' - (52)
- Path Equivalence: 'filename ' (Trailing Space) - (46)
- PHP External Variable Modification - (473)
- Reachable Assertion - (617)
- Signed to Unsigned Conversion Error - (195)
- Unbounded Transfer ('Classic Buffer Overflow') - (120)
- Unchecked Return Value - (252)
- Unchecked Return Value to NULL Pointer Dereference - (690)
- Unparsed Raw Web Content Delivery - (433)
- Unrestricted File Upload - (434)
- Use After Free - (416)
- Use of Hard-coded Cryptographic Key - (321)
- Write-what-where Condition - (123)

Named Chains

CWE-680	Integer Overflow to Buffer Overflow
CWE-690	Unchecked Return Value to NULL Pointer Dereference
CWE-692	Incomplete Blacklist to Cross-Site Scripting

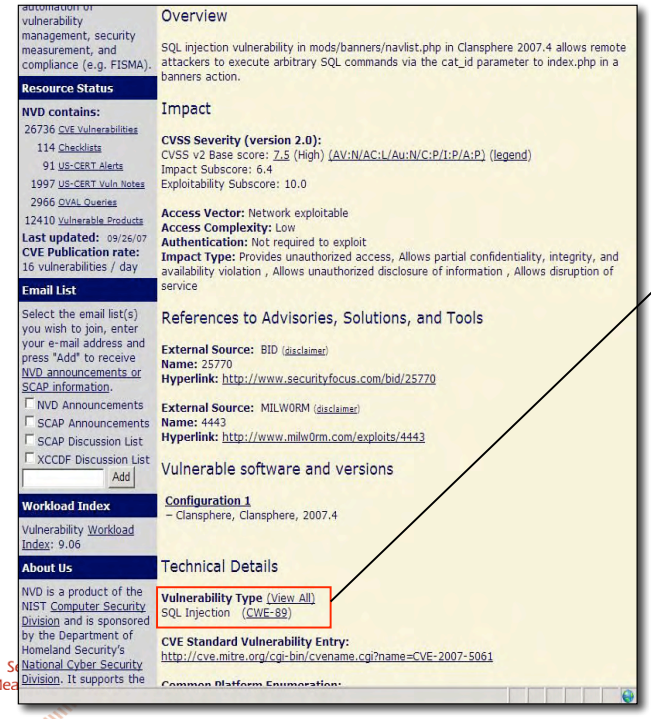
Chains

CWE-46 Path Equivalence: 'filename ' (Trailing Space)
↳ **CWE-289 Authentication Bypass by Alternate Name**
CWE-52 Path Equivalence: '/multiple/trailing/slash//'
↳ **CWE-289 Authentication Bypass by Alternate Name**
CWE-93 Failure to Sanitize CRLF Sequences (aka 'CRLF Injection')
↳ CWE-117 Incorrect Output Sanitization for Logs
CWE-113 Failure to Sanitize CRLF Sequences in HTTP Headers (aka 'HTTP Response Splitting')
↳ **CWE-79 Failure to Sanitize Directives in a Web Page (aka 'Cross-site scripting' (XSS))**
↳ CWE-494 Download of Untrusted Mobile Code Without Integrity Check
↳ CWE-692 Incomplete Blacklist to Cross-Site Scripting
CWE-171 Cleansing, Canonicalization, and Comparison Errors
↳ **CWE-289 Authentication Bypass by Alternate Name**
CWE-173 Failure to Handle Alternate Encoding
↳ **CWE-289 Authentication Bypass by Alternate Name**
CWE-178 Failure to Resolve Case Sensitivity
↳ CWE-433 Unparsed Raw Web Content Delivery
↳ **CWE-289 Authentication Bypass by Alternate Name**
CWE-184 Incomplete Blacklist
↳ **CWE-79 Failure to Sanitize Directives in a Web Page (aka 'Cross-site scripting' (XSS))**
↳ CWE-78 Failure to Sanitize Data into an OS Command (aka 'OS Command Injection')
↳ CWE-434 Unrestricted File Upload *(also a composite)*
↳ **CWE-98 Insufficient Control of Filename for Include/Require Statement in PHP Program (aka 'PHP File Inclusion')** *(also a composite)*
CWE-190 Integer Overflow (Wrap or Wraparound)
↳ **CWE-120 Unbounded Transfer ('Classic Buffer Overflow')** *(also a composite)*
↳ **CWE-123 Write-what-where Condition**
↳ CWE-680 Integer Overflow to Buffer Overflow
↳ **CWE-122 Heap-based Buffer Overflow**
CWE-193 Off-by-one Error
↳ CWE-617 Reachable Assertion
↳ CWE-170 Improper Null Termination
↳ **CWE-120 Unbounded Transfer ('Classic Buffer Overflow')** *(also a composite)*
CWE-195 Signed to Unsigned Conversion Error
↳ **CWE-122 Heap-based Buffer Overflow**
CWE-252 Unchecked Return Value
↳ CWE-476 NULL Pointer Dereference
↳ CWE-690 Unchecked Return Value to NULL Pointer Dereference
CWE-390 Detection of Error Condition Without Action
↳ CWE-401 Failure to Release Memory Before Removing Last Reference (aka 'Memory Leak')
CWE-416 Use After Free
↳ **CWE-120 Unbounded Transfer ('Classic Buffer Overflow')** *(also a composite)*
↳ **CWE-123 Write-what-where Condition**
CWE-456 Missing Initialization
↳ CWE-89 Failure to Sanitize Data into SQL Queries (aka 'SQL Injection')
↳ **CWE-120 Unbounded Transfer ('Classic Buffer Overflow')** *(also a composite)*
CWE-473 PHP External Variable Modification
↳ **CWE-98 Insufficient Control of Filename for Include/Require Statement in PHP Program (aka 'PHP File Inclusion')** *(also a composite)*
CWE-600 Failure to Catch All Exceptions (Missing Catch Block)
↳ CWE-209 Error Message Information Leaks
CWE-602 Design Principle Violation: Client-Side Enforcement of Server-Side Security
↳ CWE-471 Modification of Assumed-Immutable Data (MAID)
CWE-656 Design Principle Violation: Reliance on Security through Obscurity
↳ CWE-259 Hard-Coded Password
↳ CWE-321 Use of Hard-coded Cryptographic Key
↳ CWE-472 External Control of Assumed-Immutable Web Parameter

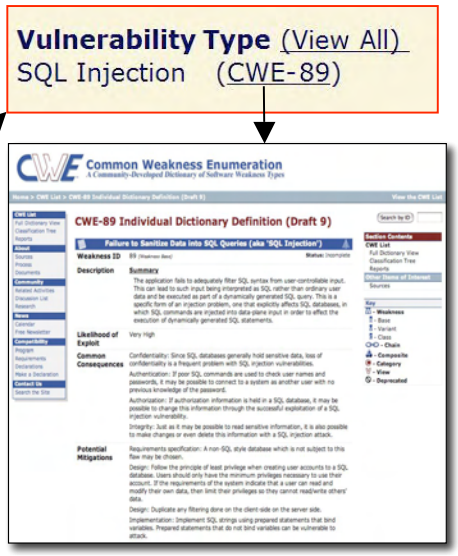


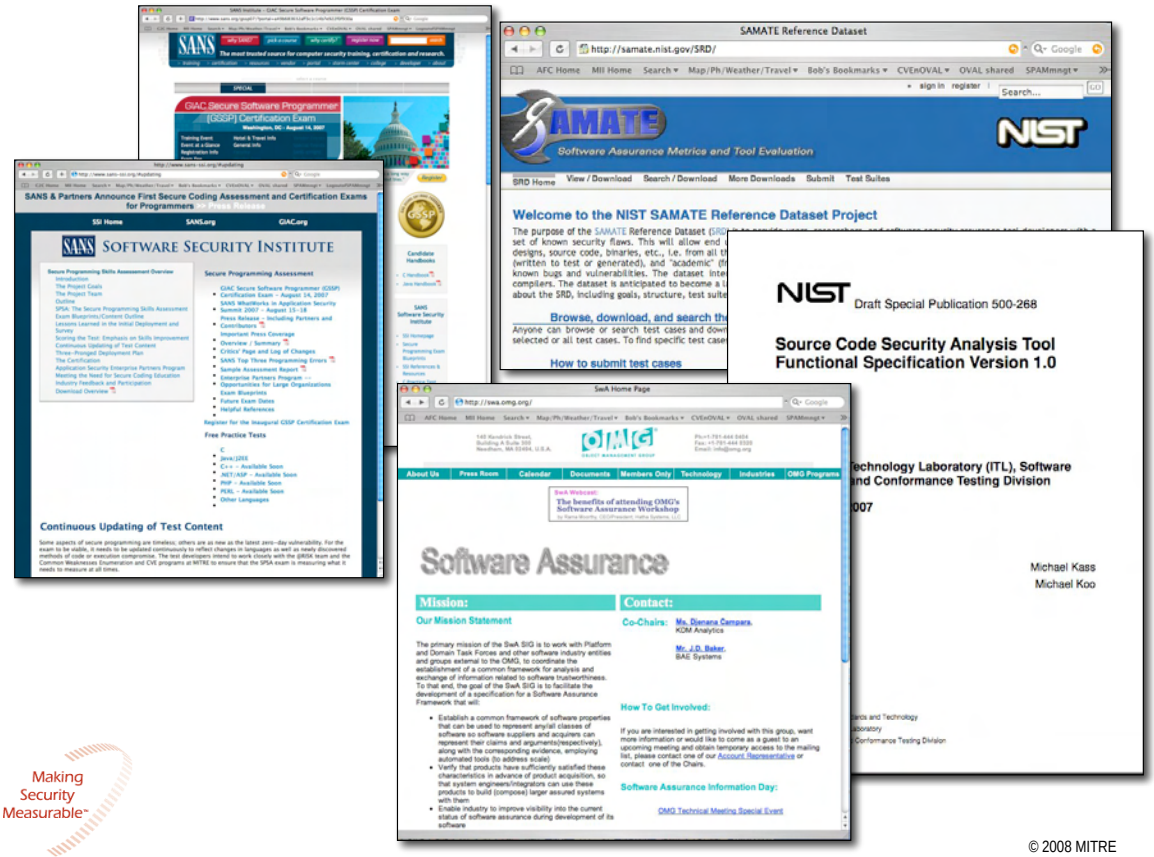
Making Security Measurable™

Some High-Level CWEs Are Now Part of the NVD CVE Information

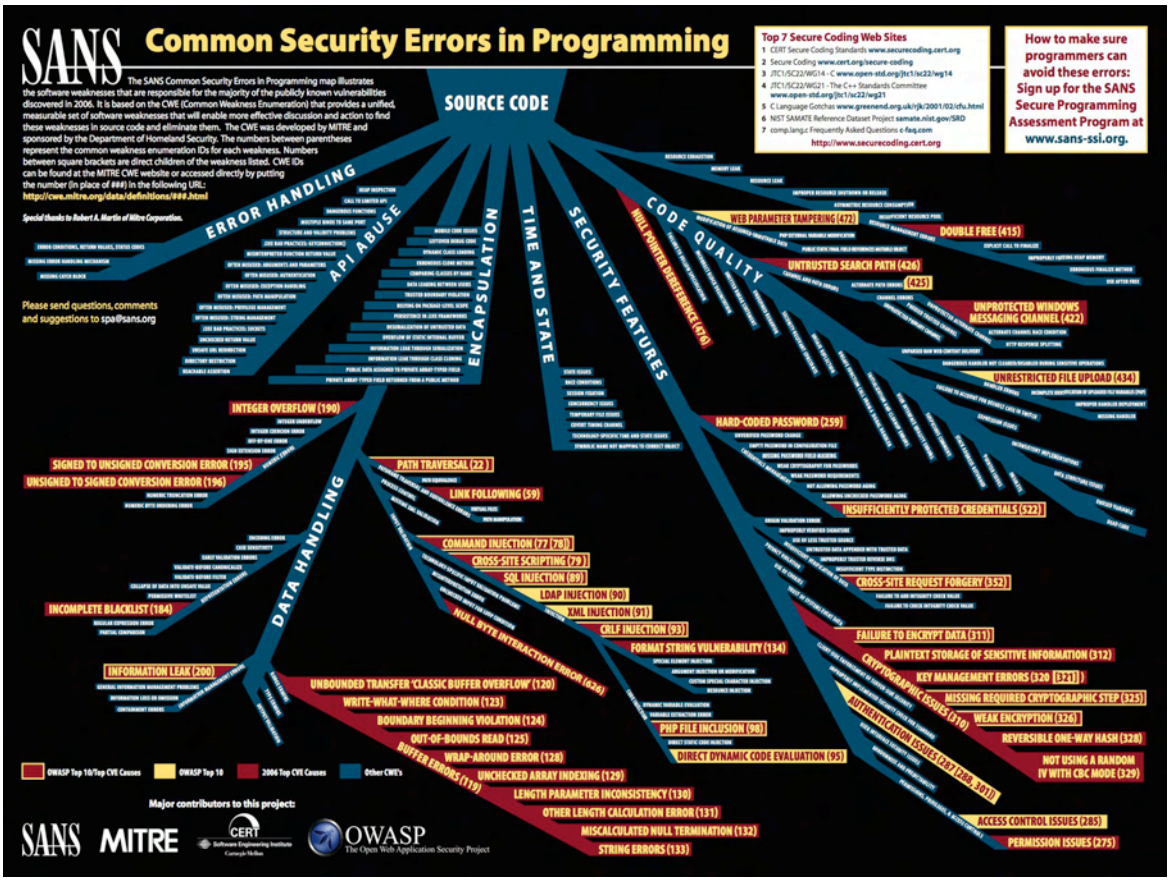


NVD XML feeds also include CWE

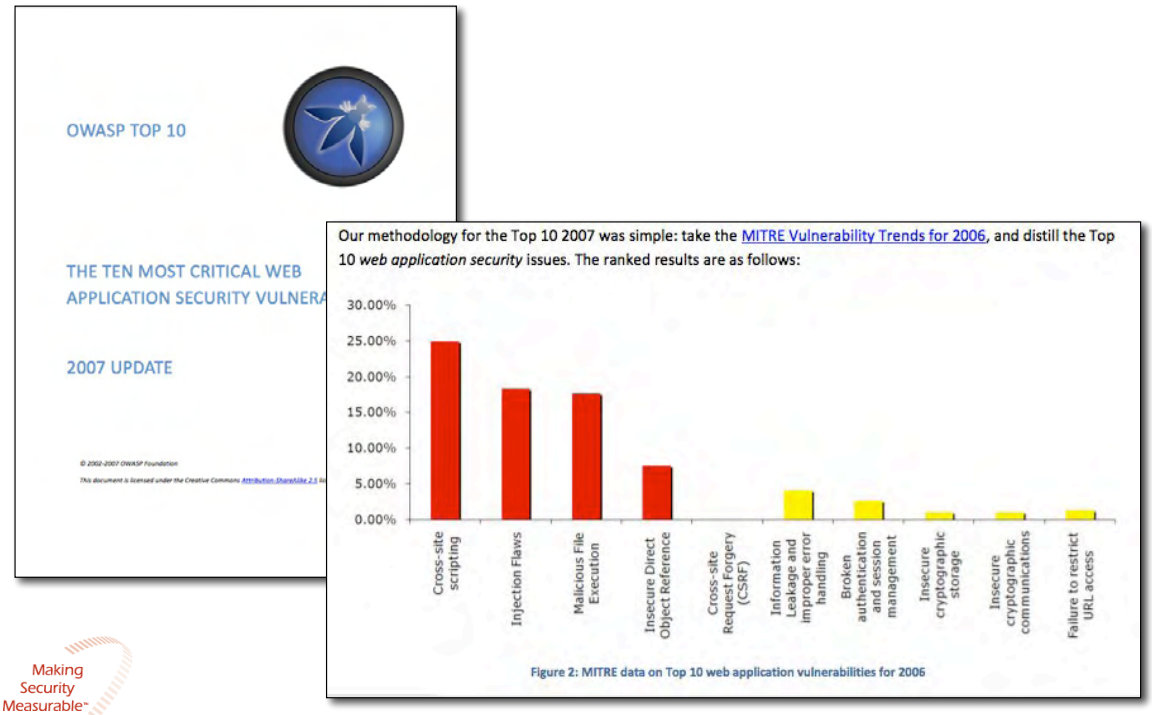




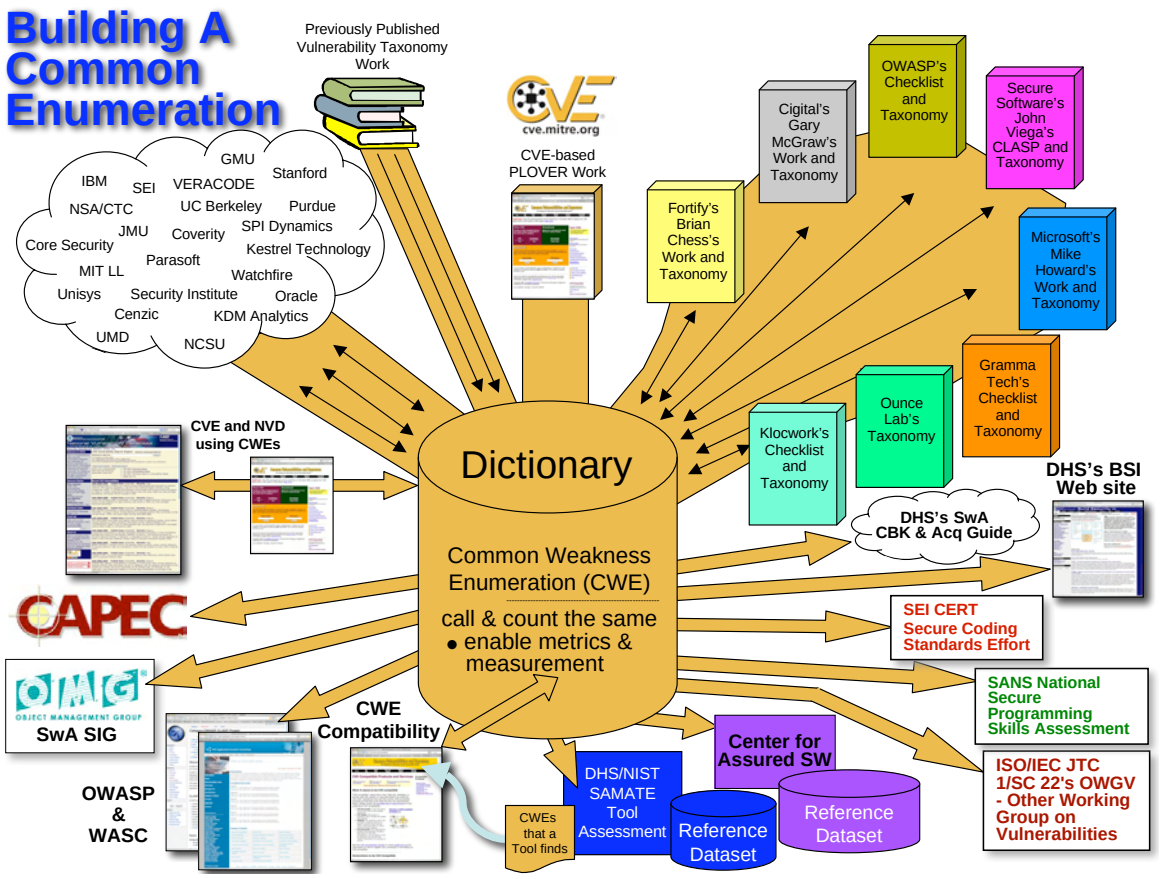
Making Security Measurable™



OWASP Top Ten 2007



Making Security Measurable™



A Resource for Creating the Attack Resistance/Resilience Testing Target List

Making Security Measurable™

Attack Patterns Overview

- Represent common approaches to attack
- Abstracted from actual exploits and attacks
- Gives you an attacker's perspective you may not have on your own
- Excellent resource for many key activities
  - **Abuse Case development**
  - **Architecture attack resistance analysis**
  - **Risk-based security testing**
  - **Red team penetration testing**
- Resources
  - **Attack Patterns article series on Build Security In website (buildsecurityin.us-cert.gov)**
  - **Common Attack Pattern Enumeration and Classification (CAPEC)**
  - ***Exploiting Software* [Hoglund & McGraw 04]**
- Primarily attack-centric testing methods

Making Security Measurable™

What is CAPEC?

- Community effort targeted at:
  - **Standardizing the capture and description of attack patterns**
  - **Collecting known attack patterns into an integrated enumeration that can be consistently and effectively leveraged by the community**
  - **Classifying attack patterns such that users can easily identify the subset of the entire enumeration that is appropriate for their context**
- Where is CAPEC today?
  - **http://capec.mitre.org**
  - **Currently 101 patterns**
  - **Future plans**
    - **New patterns**
    - **Align patterns with other resources**
    - **Formalize patterns to finer granularity to support test case generation and bridging with the malware and incident response communities**

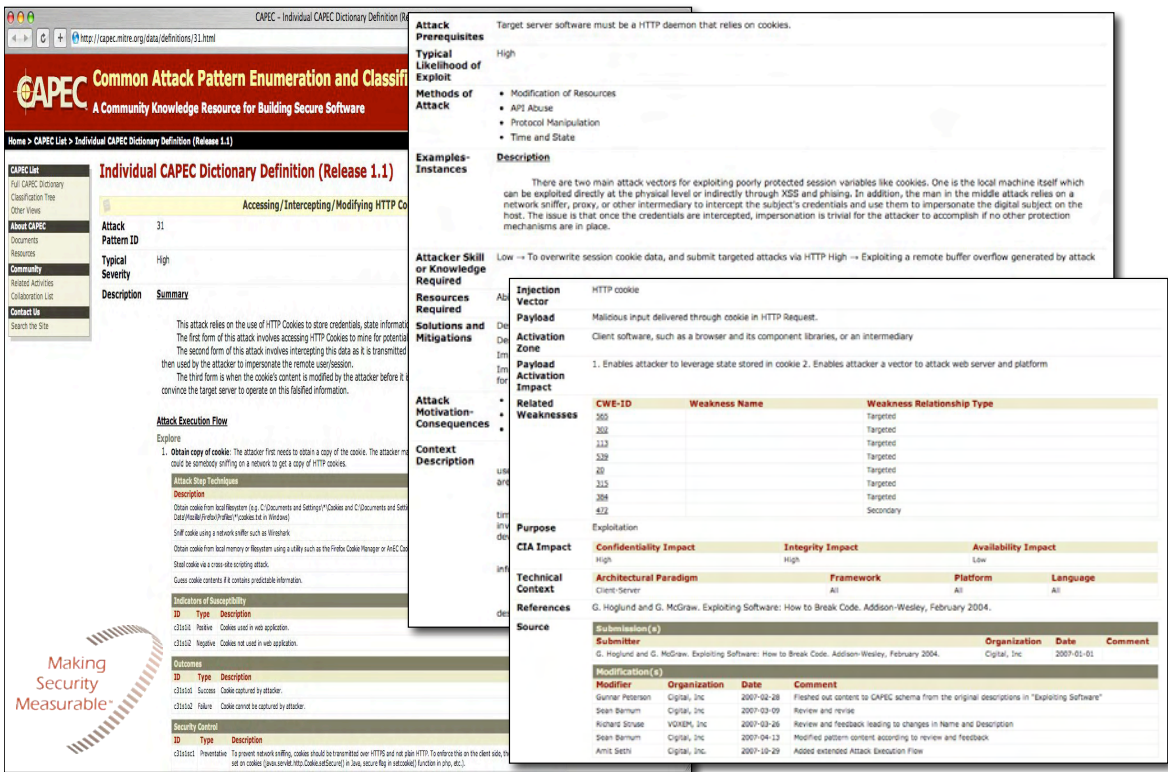
Making Security Measurable™

What do Attack Patterns Look Like?

- Primary Schema Elements
  - **Identifying Information**
    - Attack Pattern ID
    - Attack Pattern Name
  - **Describing Information**
    - Description
    - Related Weaknesses
    - Related Vulnerabilities
    - Method of Attack
    - Examples-Instances
    - References
  - **Prescribing Information**
    - Solutions and Mitigations
  - **Scoping and Delimiting Information**
    - Typical Severity
    - Typical Likelihood of Exploit
    - Attack Prerequisites
    - Attacker Skill or Knowledge Required
    - Resources Required
    - Attack Motivation-Consequences
    - Context Description
- Supporting Schema Elements
  - **Describing Information**
    - Injection Vector
    - Payload
    - Activation Zone
    - Payload Activation Impact
  - **Diagnosing Information**
    - Probing Techniques
    - Indicators-Warnings of Attack
    - Obfuscation Techniques
  - **Enhancing Information**
    - Related Attack Patterns
    - Relevant Security Requirements
    - Relevant Design Patterns
    - Relevant Security Patterns

Making Security Measurable™

Common Attack Pattern Enumeration and Classification (CAPEC)



How do you identify relevant attack patterns?

- Abuse cases from requirements
- Threat modeling as part of architectural risk analysis
- Contextual mapping

Making Security Measurable™

Leveraging attack patterns as test case templates

- Attack patterns contain information that can significantly assist in defining contexts, preconditions, test data, action steps, postconditions and variation axes for security test cases
  - **Context: Context Description, Examples-Instances, Related Weaknesses, Related Vulnerabilities, Relevant Security Requirements, Relevant Design Patterns, Relevant Security Patterns**
  - **Preconditions: Attack Prerequisites, Attacker Skill or Knowledge Required, Resources Required**
  - **Test data: Description**
  - **Action steps: Description, Method of Attack, Injection Vector, Payload, Activation Zone**
  - **Postconditions: Description, Attack Motivation-Consequences, Payload Activation Impact**
  - **Variation axes: Description, Solutions and Mitigations, Probing Techniques, Obfuscation Techniques**

Making Security Measurable™

Very simplistic test case example

Test Case 1: Single quote SQL injection of registration page web form fields

Test Case Goal: Ensure SQL syntax single quote character entered in registration page web form fields does not cause abnormal SQL behavior

Context:

- This test case is part of a broader SQL injection syntax exploration suite of tests to probe various potential injection points for susceptibility to SQL injection. If this test case fails, it should be followed-up with test cases from the SQL injection experimentation test suite.

Preconditions:

- Access to system registration page exists
- Registration page web form field content are used by system in SQL queries of the system database upon page submission
- User has the ability to enter free-form text into registration page web form fields

Test Data:

- ASCII single quote character

Action Steps:

- Enter single quote character into each web form field on the registration page
- Submit the contents of the registration page

Postconditions:

- Test case fails if SQL error is thrown
- Test case passes if page submission succeeds without any SQL errors

Making Security Measurable™

Attack Pattern Schema Formalization: Improving the value for test case generation

- Current effort underway to provide a more formalized schema for the attack pattern Description element to better support test case definition and eventually automated generation
- First step of this formalization design has been completed and 25 of the 101 CAPEC attack patterns have been updated to be compliant
- Future work will involve updating the rest of the CAPEC attack patterns and diving into even deeper levels of formalization targeted at supporting automation

Making Security Measurable™

Attack Pattern Description Schema Formalization

Description

- **Summary**
- **Attack_Execution_Flow**
  - **Attack_Phase$^{1..3}$ (Name(Explore, Experiment, Exploit))**
    - **Attack_Step$^{1..*}$**
      - **Attack_Step_Title**
      - **Attack_Step_Description**
      - **Attack_Step_Technique $^{0..*}$**
        - » **Attack_Step_Technique_Description**
        - » **Environments**
      - **Indicator$^{0..*}$ (ID, Type(Positive, Failure, Inconclusive))**
        - » **Indicator_Description**
        - » **Environments**
      - **Outcome$^{0..*}$ (ID, Type(Success, Failure, Inconclusive))**
      - **Security Control$^{0..*}$ (ID, Type(Detective, Corrective, Preventative))**

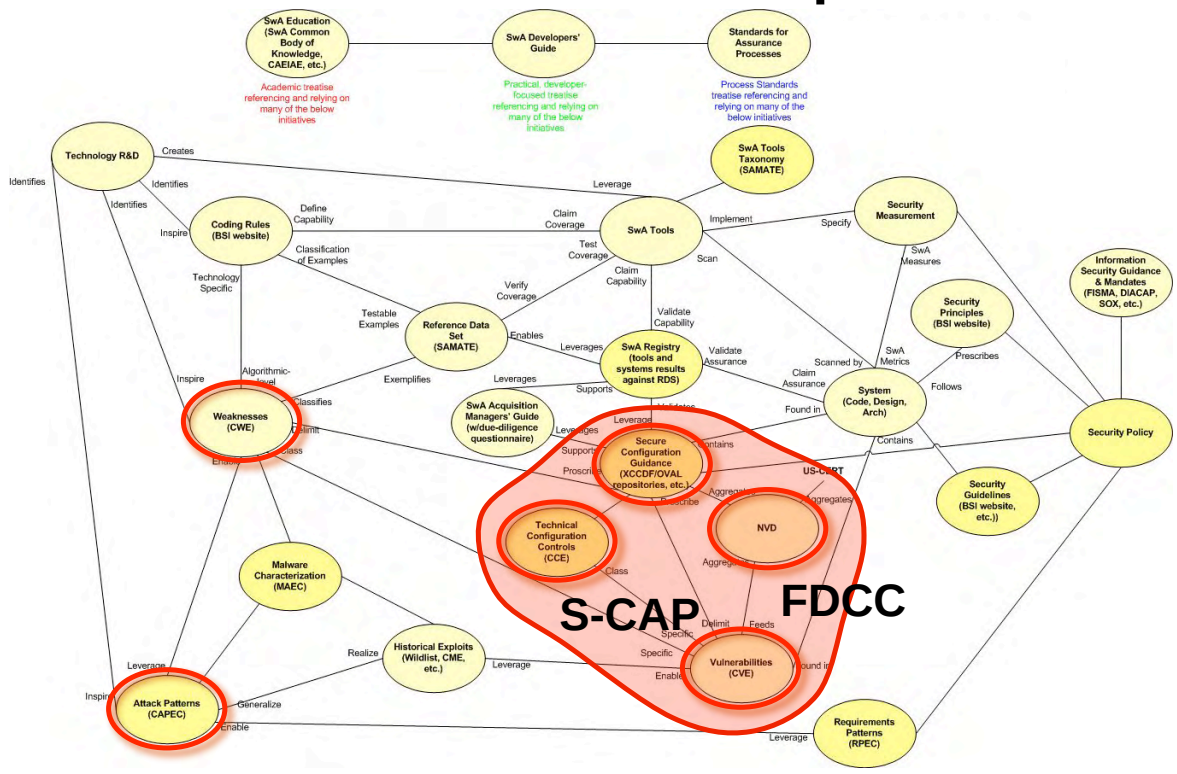
Making Security Measurable™

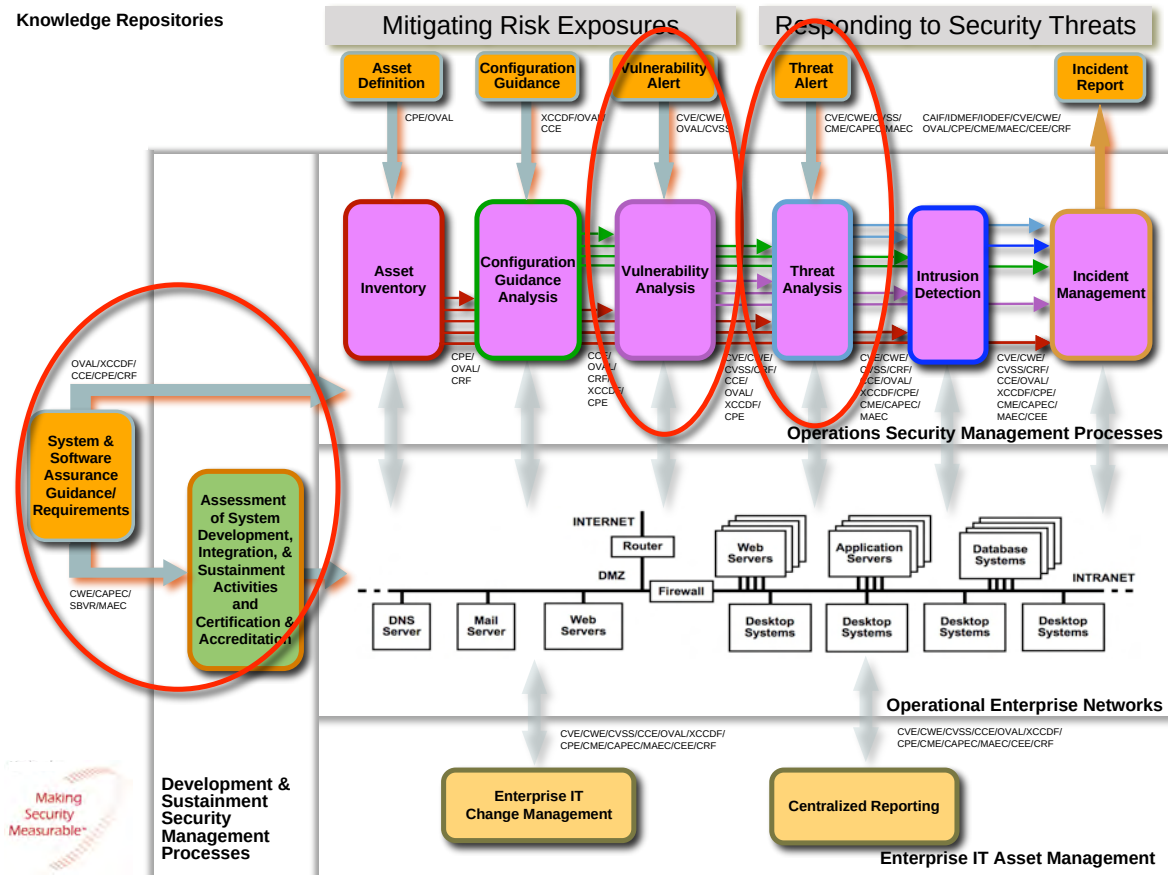
Leveraging the formalized schema

- Modular test cases should be defined for each Attack_Phase
- Attack_Step enumeration forms the foundation for test case action steps
- Equivalence classes should be defined around each variation axes including Attack_Step, Attack_Technique & Security Control
- Indicators and Outcomes should be used for defining test case postconditions
- Environments should be used for defining test case contexts and preconditions

Making Security Measurable™

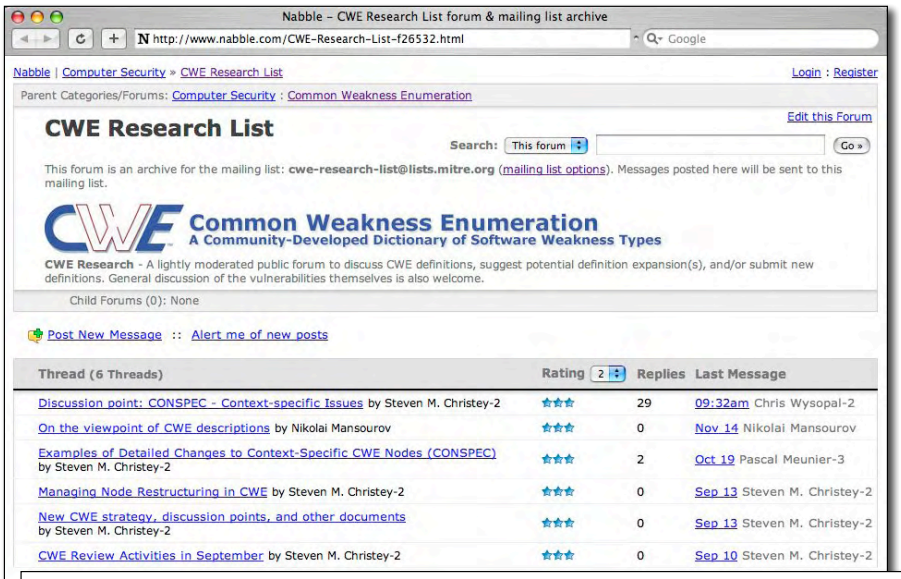
Software Assurance Programs and Activities Landscape





Making Security Measurable™

[makingsecuritymeasurable.mitre.org]

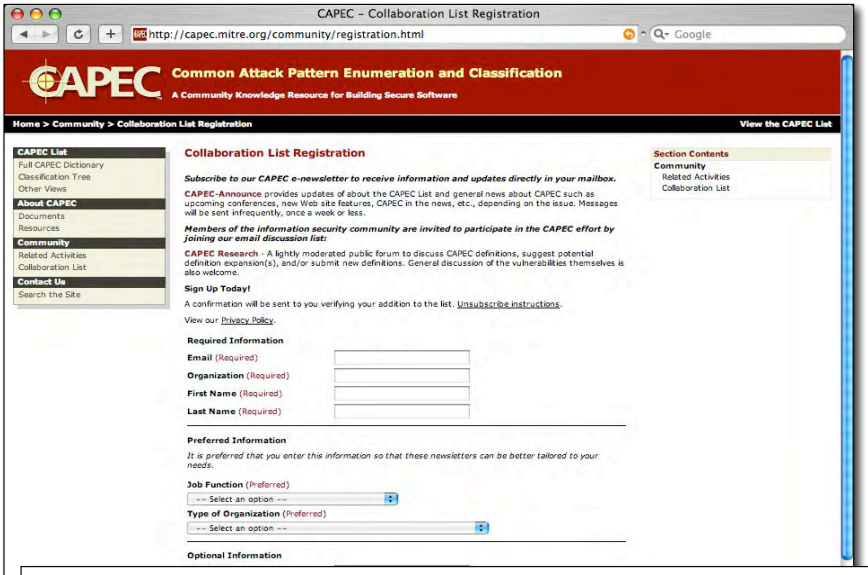


To subscribe, see:

http://cwe.mitre.org/community/registration.html

or just send an email to listserv@lists.mitre.org with the command:
subscribe CWE-RESEARCH-LIST

Making Security Measurable™



To subscribe, see:

http://capec.mitre.org/community/registration.html

or just send an email to listserv@lists.mitre.org with the command:
subscribe CAPEC-RESEARCH-LIST

Making Security Measurable™

Acronyms

BSI	Build Security In
BUF	Buffer Errors
CAIF	Common Announcement Interchange Format
CAPEC	Common Attack Pattern Enumeration and Characterization
CBK	Common Body of Knowledge
CCE	Common Control Enumeration
CEE	Common Event Expression
CIS	Center for Internet Security
CLASP	Comprehensive Lightweight Application Security Process
CME	Common Malware Enumeration
CPE	Common Package Enumeration
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
CWE	Common Weakness Enumeration
DHS	Department of Homeland Security
DIACAP	Department of Defense Information Assurance Certification and Accreditation Process
DoD	Department of Defense
DOT	Relative Path Traversal Errors
DISA	Defense Information Systems Agency
eMASS	Enterprise Mission Assurance Support System
FDCC	Federal Desktop Core Configuration
FIDEF	Forensic Investigation Description and Exchange Formats
FISMA	Federal Information
IA	Information Assurance
IAVA	Information Assurance Vulnerability Assessment
IODEF	Incident Object Description and Exchange Formats
IT	Information Technology
MAEC	Malware Attribute Enumeration and Characterization
NIST	National Institute of Science and Technology
NSA	National Security Agency
NVD	National Vulnerability Database
OMB	Office of Management and Budget
OMG	Object Management Group
OSD	Office of the Secretary of Defense
OVAL	Open Vulnerability and Assessment Language
OWASP	Open Web Application Security Program
PLOVER	Preliminary List Of Vulnerability Examples for Researchers
SAMATE	Software Assurance Metrics
SBVR	Semantic Business Vocabulary and Rules
SCAP	Security Content Automation Protocols
SIM	Security Information Manager
STIGs	Security Technical Implementation Guides
SwA	Software Assurance
US-CERT	United States Computer Emergency Response Team
VEDEF	Vulnerability & Exploit Description and Exchange Formats
WASC	Web Application Security Consortium
XCCDF	eXtensible Configuration Checklist Document Format
XSS	Cross-Site Scripting
XML	eXtensible Markup Language