

Having a Defined Target for Software Security Testing



Robert A. Martin



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



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



© 2008 MITRE

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



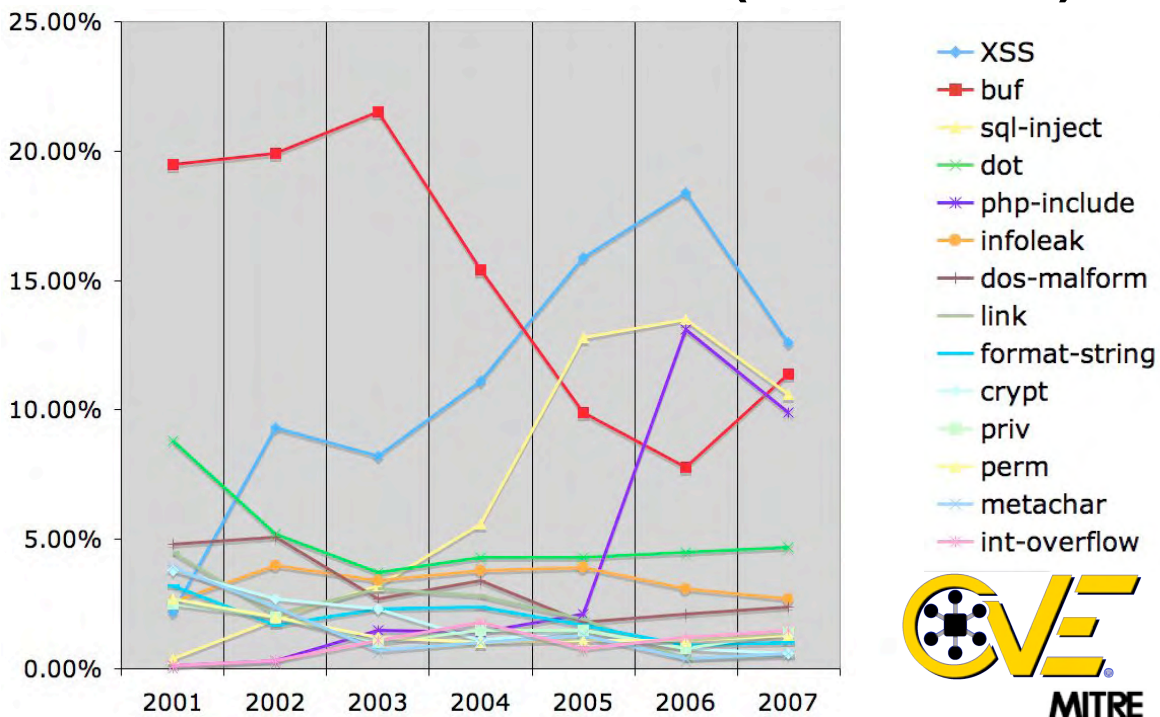
© 2008 MITRE

A Resource for Creating the Weakness/Vulnerability Testing Target List



© 2008 MITRE

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



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)



© 2008 MITRE

XSS is possible anywhere scripting is supported

- `<script ...>`
- ``
- CSS styles
- `` 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 `` examples.

© 2008 MITRE

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



© 2008 MITRE

Blacklists and XSS

- “I’ll just strip uses of <SCRIPT>”
 - **This works:** ``
- “But I want to support IMG tags, so I’ll just strip ‘javascript’”
 - **This works (no lie):** ``
- “I’ll decode everything, THEN look for ‘javascript’”
 - **This works (no lie):** ``
- “I’ll make sure that only ‘http’ is allowed”
 - **This works:** ``
- “I’ll make sure to strip out ‘onmouseover’”
 - **This works:** ``
- “I’ll only support SRC for IMG tags”
 - **This works (no lie):** `<b onmouseover="javascript:alert('hi')">hello`
- ... and many, many more



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.

© 2008 MITRE

Integer Overflows

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

- Frequently bypassed in signed comparison



© 2008 MITRE

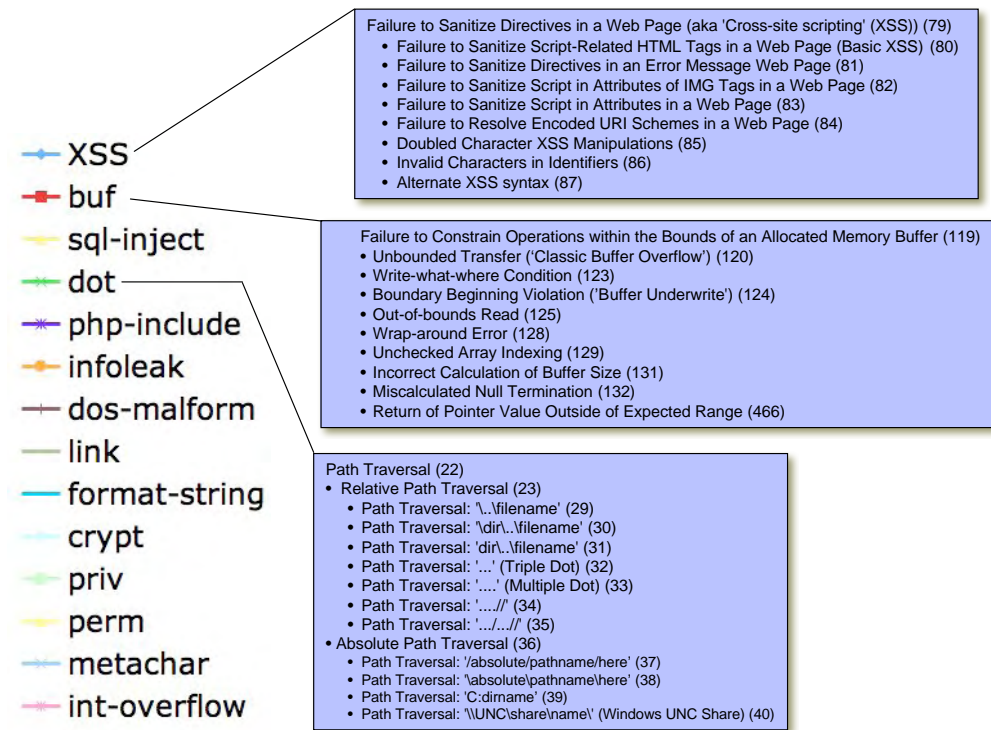
Buffer Overflow Variants

- 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**



© 2008 MITRE

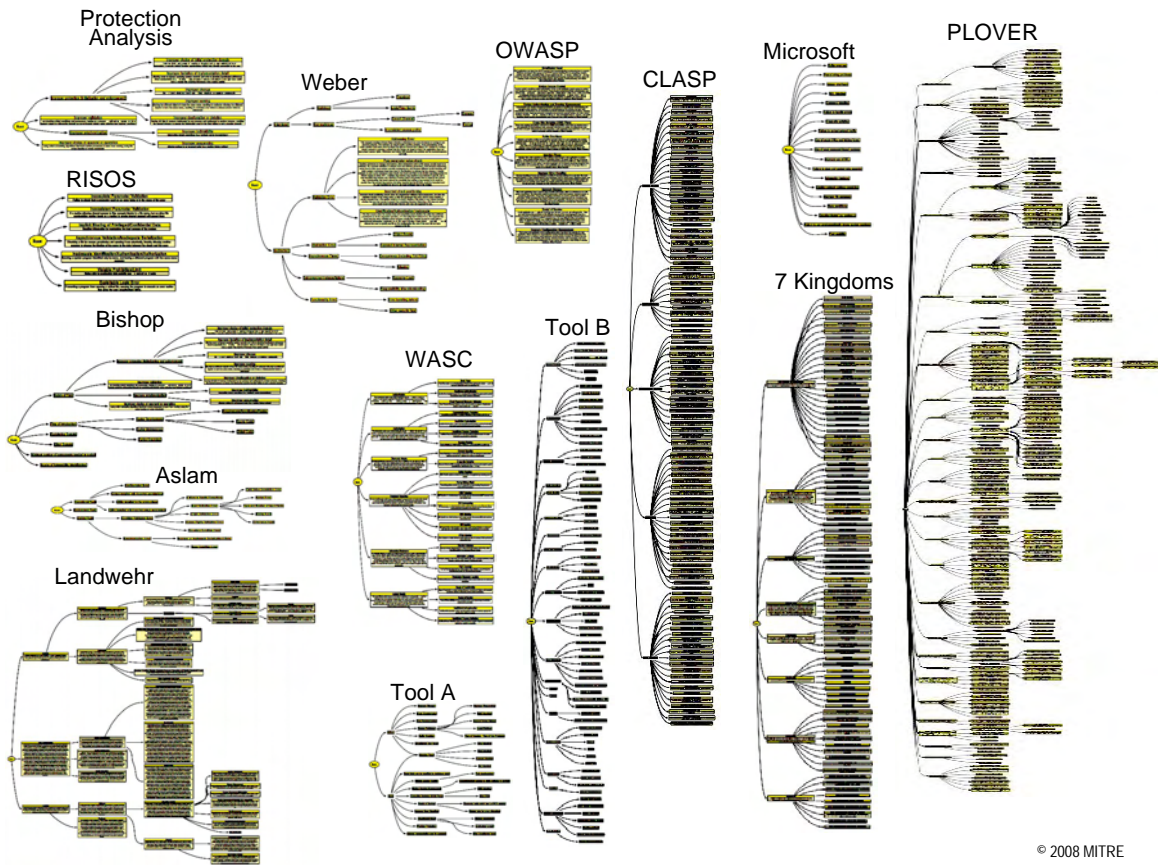
Removing and Preventing the Vulnerabilities Requires More Specific Definitions...CWEs



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**





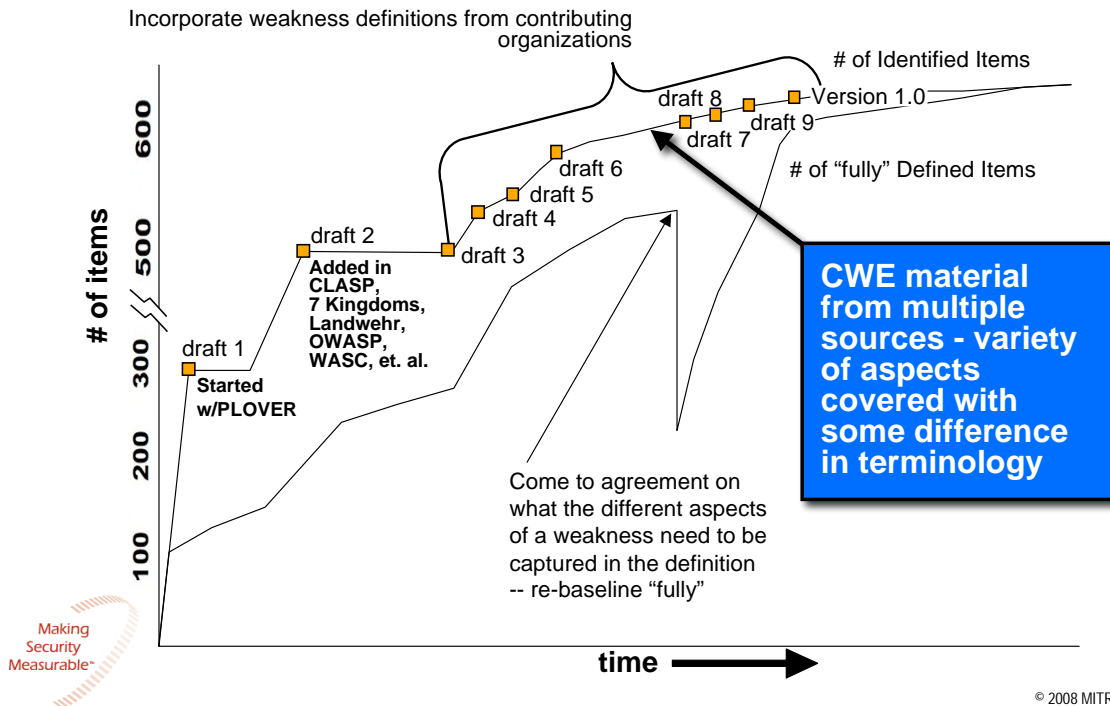
Using A Unilateral NDA with MITRE to Bring in Info

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



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

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	Relationship_Chain	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				

[BACK TO TOP](#)

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 "CWE*", 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

[BACK TO TOP](#)

Views

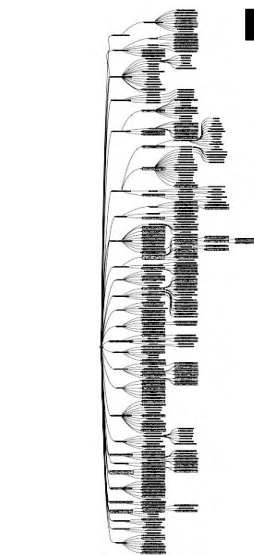
The Views structure contains zero or more View elements. Each View element represents a perspective with which one might look at the weaknesses in CWE. CWE-630 Weaknesses Examined by SAMATE and CWE-658 Weaknesses found in the C Language are two examples of Views.

[BACK TO TOP](#)

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.

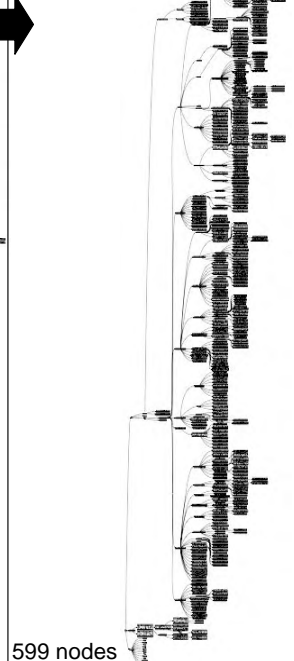
PLOVER



300 nodes

2005

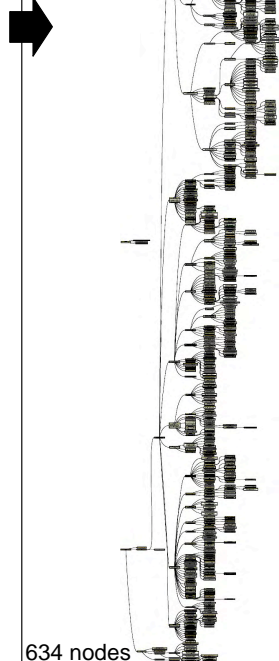
CWE
draft 5



599 nodes

2006

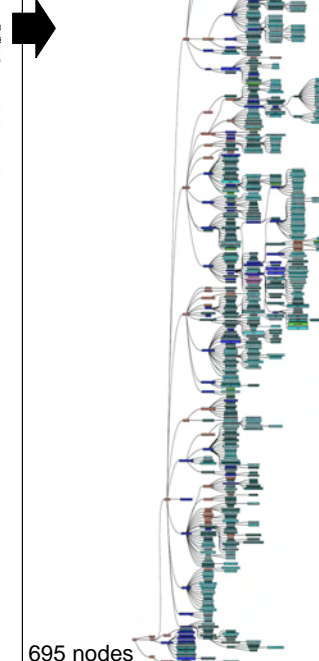
CWE
draft 7



634 nodes

2007

CWE
draft 9



695 nodes

4/11/2008

© 2008 MITRE

CWE Draft 9 (11 April 08)

VIEW SLICE: CWE-2000: Comprehensive CWE Dictionary (Draft 9)

Comprehensive CWE Dictionary

View ID: 2000 (view)

Objective: This view (slice) covers all the elements in CWE.

View Filter Used: true!

CWEs in this view	Total CWEs
Total	695 out of 695
Views	14 out of 14
Categories	64 out of 64
Weaknesses	605 out of 605
Compound Elements	12 out of 12

Absolute Path Traversal

Weakness ID 35 (Weakness Base)

Description Summary: The software, when constructing file or directory names from input, does not properly sanitize absolute path sequences, such as "/path/here".

Potential Mitigations: see "Path Traversal" (CWE-22)

Relationships

Nature	Type	ID	Name
ChildOf	WE	22	Path Traversal
ParentOf	WE	37	Path Traversal: Absolute Pathname Check
ParentOf	WE	38	Path Traversal: Absolute Pathname Check

Making Security Measurable™

CWE-119 Individual Dictionary Definition (Draft 9)

Failure to Constrain Operations within the Bounds of an Allocated Memory Buffer

Weakness ID 119 (Weakness Base) Status: Draft

Description Summary: The software may potentially allow operations, such as reading or writing, to be performed at addresses not intended by the developer.

Extended Description: When software permits read or write operations on memory located outside of an allocated range, an attacker may be able to access/modify sensitive information, cause the system to crash, alter the intended control flow, or execute arbitrary code.

Affected Resource: Memory

Relationships

Nature	Type	ID	Name
ChildOf	WE	118	Range Errors
ChildOf	WE	635	Weaknesses Used by NVD
ChildOf	WE	633	Weaknesses that Affect Memory
ParentOf	WE	133	String Errors
ParentOf	WE	123	Write-what-where Condition
ParentOf	WE	124	Boundary Beginning Violation ("Buffer Underwrite")
ParentOf	WE	125	Out-of-bounds Read
ParentOf	WE	128	Wrap-around Error
ParentOf	WE	129	Unchecked Array Indexing
ParentOf	WE	131	Incorrect Calculation of Buffer Size
ParentOf	WE	132	Miscalculated Null Termination
ParentOf	WE	466	Return of Pointer Value Outside of Expected Range
ParentOf	WE	120	Unbounded Transfer ("Classic Buffer Overflow")

Related Attack Patterns

CAPEC-ID	Attack Pattern Name
100	Overflow Buffers
10	Buffer Overflow via Environment Variables
14	Client-side Injection-induced Buffer Overflow
42	MMIO Conversion
24	Filter Failure through Buffer Overflow
8	Buffer Overflow in an API Call
44	Overflow Binary Resource File
9	Buffer Overflow in Local Command-Line Utilities
45	Buffer Overflow via Symbolic Links
46	Overflow Variables and Tags
47	Buffer Overflow via Parameter Expansion

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

Term	Attack	Vuln/Weakness	Consequence
Buffer Overflow	<ul style="list-style-type: none"> Long string argument Length field inconsistency Large number of events, etc. 	<ul style="list-style-type: none"> Failure to restrict length Failure to control offset Error in attempting to do either of these 	<ul style="list-style-type: none"> 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)	<ul style="list-style-type: none"> Failure to fully control contents of format strings 	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> Provide invalid argument Monitor behavioral or timing results Sniff 	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> <<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	<ul style="list-style-type: none"> Execution of script code Modification of format or presentation
DoS	Provide invalid argument	<ul style="list-style-type: none"> Failure to anticipate or handle error conditions Failure to properly limit scope of an error 	<ul style="list-style-type: none"> Crash "Memory Corruption" Infinite loop
DoS	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> Crash "Memory Corruption" Infinite loop
Authentication Bypass	<ul style="list-style-type: none"> Perform invalid sequence of instructions, e.g. direct request Replay challenge/response Cookie modification SQL injection 	<ul style="list-style-type: none"> 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

Differences between Draft 8 and Draft 9

Total new	39
Total deprecated	1
Total shared	656
Total important changes	429
Total major changes	463
Total minor changes	399
Total minor changes (no major)	134
Total unchanged	59

Any change with respect to whitespace is ignored. "Minor" changes are text changes that only affect capitalization and punctuation. Most other changes are marked as "Major." Simple schema changes are treated as Minor, such as the change from AffectedResource to Affected_Resource in Draft 8, or the relationship name change from "CanResultIn" to "CanPrecede" in Draft 9. For each mutual relationship between nodes A and B (such as ParentOf and ChildOf), a relationship change is noted for both A and B.

Field	Major	Minor
Affected_Resource	1	0
Alternate_Terms	1	0
Applicable_Platforms	2	0
Black_Box_Definition	0	0
CVEs_Mentioned	3	0
Causal_Nature	0	0
Common_Consequences	3	0
Common_Methods_of_Exploitation	0	0
Context_Notes	0	0
Demonstrative_Example	186	50
Description	186	50
Detection_Factor	0	0
Enabling_Factors_for_Exploitation	0	0
Functional_Area	0	0
Likelihood_of_Exploit	0	0
Name	248	1
Node_Relationship	202	71
Observed_Example	11	1
Potential_Mitigations	10	0
References	3	0
Related_Attack_Patterns	0	0
Relevant_Properties	0	0
Research_Gaps	2	0
Source_Taxonomy	59	0
Time_of_Introduction	0	0
Type	24	377
Weakness_Ordinality	2	0
White_Box_Definition	0	0

CWE "Scrub"

CWE Working Documents

Short-Term Strategy for CWE Community Feedback

For Fall 2007, MITRE's short-term strategy for obtaining CWE community feedback is as follows.

Types of CWE Modifications

CWE modifications can occur at three different levels, based on their overall impact on CWE and its consumers.

- Systemic Modifications:** require active feedback from the community, because they affect many stakeholders and could force extensive modifications to a large number of CWE nodes. They could cause many nodes to be merged or split compared to the current CWE versions; other kinds of nodes might be excluded entirely.
- Major Modifications:** affect multiple nodes, but are localized to only one portion of CWE (such as path traversal and its variants) or involve significant additions or modifications (such as recording new relationships for a view, or changing the meaning of a particular field). Some discussion with the community would be fruitful to both the community and CWE.
- Minor Modifications:** small and localized, only affecting single nodes, such as: fixing typos and grammar errors, changing the name, clarifying the description and related commentary, filling out blank fields, providing examples, etc. Little or no discussion with the community is needed, although feedback is always welcome.

These modifications will have mixed priorities. For example, some minor modifications might be treated as high priority by MITRE, such as nodes that do not have descriptions, or changes that have active interest by our sponsors or the community at large.

General Community Review Process

1. Identify stakeholders and invite to participate in CWE Researchers List
2. Document the Systemic Issues that could impact a significant portion of CWE.
3. Propose these Systemic Issues as "discussion points" to the Researcher list.
4. Manage community feedback.
5. Make final decisions regarding the discussion points.
6. Determine required schema changes.
7. Modify CWE nodes according to final decisions from step 5.
8. Make other high-priority edits to CWE nodes based on prioritization and community interest.
9. Release incremental drafts for each significant gain.
10. Repeat steps 3 to 9 as needed, until CWE becomes stable.
11. Add new nodes.
12. Release CWE Version 1.0.
13. Further extend the CWE Community as needed.

Milestones

Dates are estimated.

- Complete: Identify stakeholders
- September 13: Finish documentation and publication of Systemic issues.
- September 17: Identify and define at least 2 views.
- September 13-24: engage community on at least 2 Systemic Issues (discussion points) and manage feedback.
- September 13-24: make high-priority Minor edits to CWE nodes that are not likely to be affected by Systemic changes.
- September 17-21: Perform CWE schema modification.

Short-Term Strategy for CWE Community Feedback

For Fall 2007, MITRE's short-term strategy for obtaining CWE community feedback is as follows.

Types of CWE Modifications

CWE modifications can occur at three different levels, based on their overall impact on CWE and its consumers.

- Systemic Modifications:** require active feedback from the community, because they affect many stakeholders and could force extensive modifications to a large number of CWE nodes. They could cause many nodes to be merged or split compared to the current CWE versions; other kinds of nodes might be excluded entirely.
- Major Modifications:** affect multiple nodes, but are localized to only one portion of CWE (such as path traversal and its variants) or involve significant additions or modifications (such as recording new relationships for a view, or changing the meaning of a particular field). Some discussion with the community would be fruitful to both the community and CWE.
- Minor Modifications:** small and localized, only affecting single nodes, such as: fixing typos and grammar errors, changing the name, clarifying the description and related commentary, filling out blank fields, providing examples, etc. Little or no discussion with the community is needed, although feedback is always welcome.

These modifications will have mixed priorities. For example, some minor modifications might be treated as high priority by MITRE, such as nodes that do not have descriptions, or changes that have active interest by our sponsors or the community at large.

General Community Review Process

1. Identify stakeholders and invite to participate in CWE Researchers List
2. Document the Systemic Issues that could impact a significant portion of CWE.
3. Propose these Systemic Issues as "discussion points" to the Researcher list.
4. Manage community feedback.
5. Make final decisions regarding the discussion points.
6. Determine required schema changes.
7. Modify CWE nodes according to final decisions from step 5.
8. Make other high-priority edits to CWE nodes based on prioritization and community interest.
9. Release incremental drafts for each significant gain.
10. Repeat steps 3 to 9 as needed, until CWE becomes stable.
11. Add new nodes.
12. Release CWE Version 1.0.
13. Further extend the CWE Community as needed.

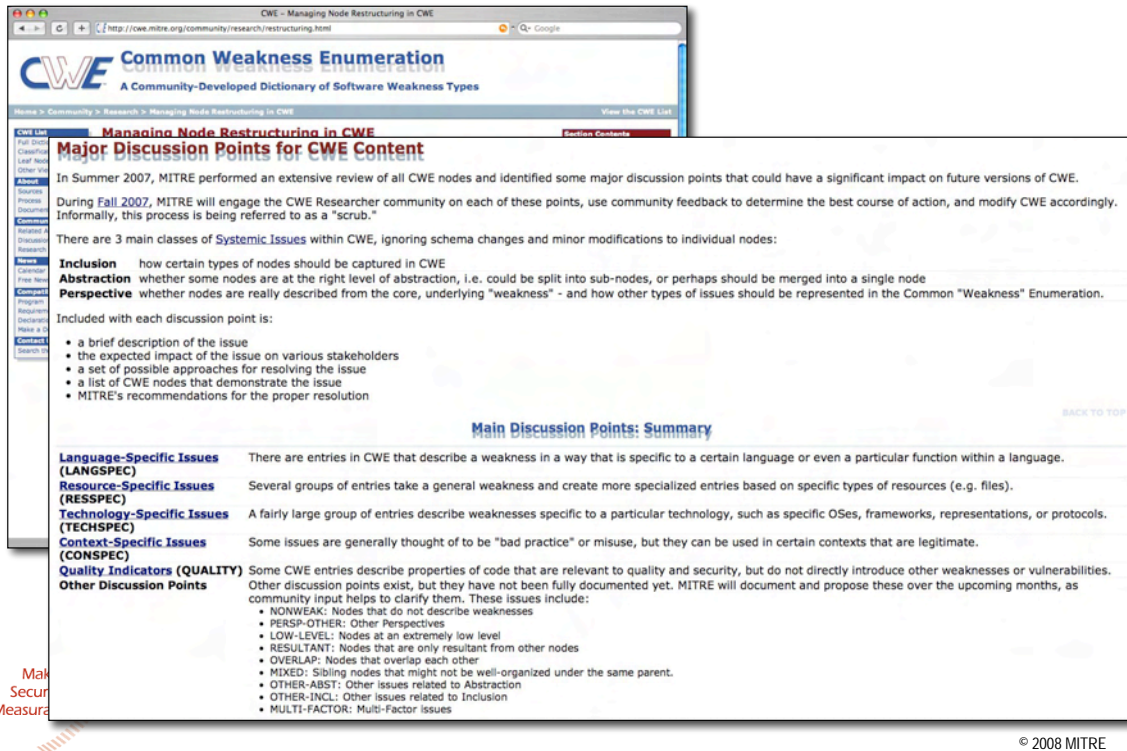
Milestones

Dates are estimated.

- Complete: Identify stakeholders
- September 13: Finish documentation and publication of Systemic issues.
- September 17: Identify and define at least 2 views.
- September 13-24: engage community on at least 2 Systemic Issues (discussion points) and manage feedback.
- September 13-24: make high-priority Minor edits to CWE nodes that are not likely to be affected by Systemic changes.
- September 17-21: Perform CWE schema modification.



CWE “Scrub”



The screenshot shows a web browser displaying the CWE website. The page title is "Managing Node Restructuring in CWE" and the main heading is "Major Discussion Points for CWE Content". The content discusses a review of CWE nodes and identifies three main classes of Systemic Issues: Inclusion, Abstraction, and Perspective. It also lists various discussion points such as Language-Specific Issues, Resource-Specific Issues, Technology-Specific Issues, Context-Specific Issues, Quality Indicators (QUALITY), and Other Discussion Points. A sidebar on the left contains navigation links like "CWE List", "Full DTD", "Classifiers", "Leaf Node", "Other Site", "About", "Sources", "Process", "Documents", "CWE List", "Research", "News", "Calendar", "FAQ", "New", "Contact", and "Search".

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

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: <ul style="list-style-type: none">• 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

© 2008 MITRE

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](#))



CWE - CWE Stakeholder Analysis

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

CWE Common Weakness Enumeration

A Community-Developed Dictionary of Software Weakness Types

Home > Community > Research > CWE Stakeholder Analysis

CWE Stakeholder Analysis

CWE Usage Scenarios Stakeholder Analysis Views

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™

© 2008 MITRE

CWE - CWE Field Completeness Goals

http://cwe-dev1.mitre.org/data/reports/field_completeness_goals.html

CWE Common Weakness Enumeration

A Community-Developed Dictionary of Software Weakness Types

Home > CWE List > Reports > CWE Field Completeness Goals

CWE Field Completeness Goals

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

Making Security Measurable™

© 2008 MITRE

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.

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	Nice	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	Nice	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	Imp	Nice	Nice	Nice	Imp	-
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	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	-	Nice	Nice	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	599	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 (0%)
Alternate_Terms	22 (8%)	8 (3%)	4 (7%)	0 (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 (0%)
Affected_Resource	22 (8%)	18 (6%)	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%)	0 (0%)	0 (0%)
Enabling_Factors_for_Exploitation	8 (3%)	7 (2%)	0 (0%)	0 (0%)
Time_of_Introduction	7 (2%)	7 (2%)	0 (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%)	0 (0%)
Black_Box_Definition	0 (0%)	0 (0%)	0 (0%)	0 (0%)

CWE - CWE Usage Scenarios

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

CWE 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

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

[BACK TO TOP](#)

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

Section Contents

- Community
- Related Activities
- Discussion List
- CWE Research**
- Discussion List Archives
- Working Documents

Making Security Measurable™

© 2008 MITRE

CWE - CWE List (Draft 9)

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

CWE Common Weakness Enumeration

A Community-Developed Dictionary of Software Weakness Types

Home > CWE List (Draft 9) 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 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.

[Search by ID](#)

Comprehensive Views

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

[BACK TO TOP](#)

Graphs

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

[BACK TO TOP](#)

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

[BACK TO TOP](#)

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

[BACK TO TOP](#)

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

Page Last Updated: April 11, 2008

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

Section Contents

- CWE List
- Full Dictionary View
- Classification Tree
- Reports
- Other Items of Interest
- Sources

CWE Views

CWE Usage Scenarios Stakeholder Analysis Views

User Issues/Questions

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

Types of Views

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

Views

V1 Programming language-specific
When programming or analyzing specific language issues of which you should be aware. Also, related characteristics.

V2 Platform-specific
When a program is run on a platform (Windows 32/64 bit, multi-processor), there are certain additions 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 know in the same place where the result is seen, or result from a weakness is useful to support 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 mistake?

Making Security Measurable™

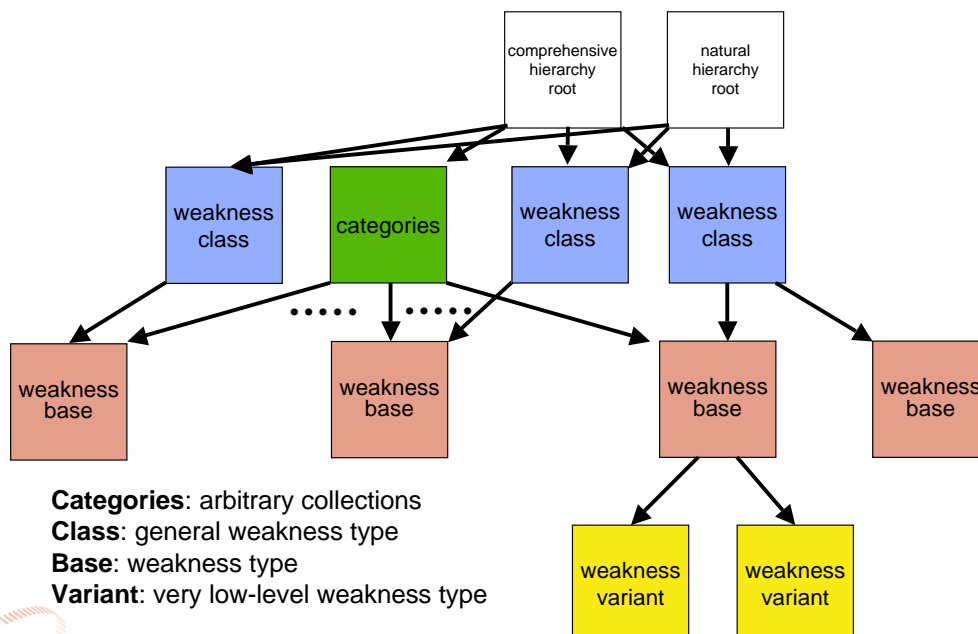
CWE-1000: Natural Hierarchy

- 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



© 2008 MITRE

Node Types

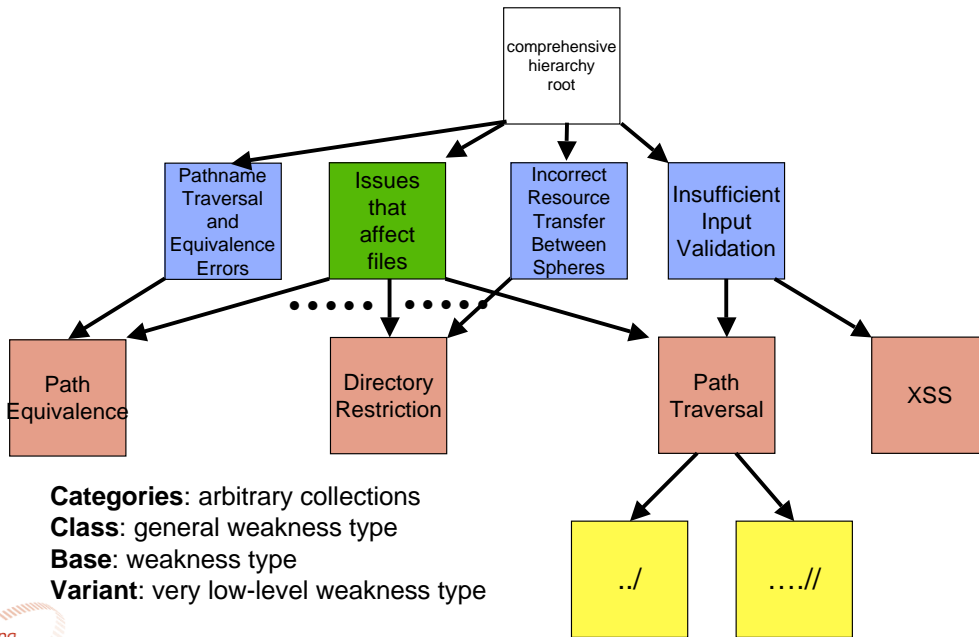


Categories: arbitrary collections
Class: general weakness type
Base: weakness type
Variant: very low-level weakness type

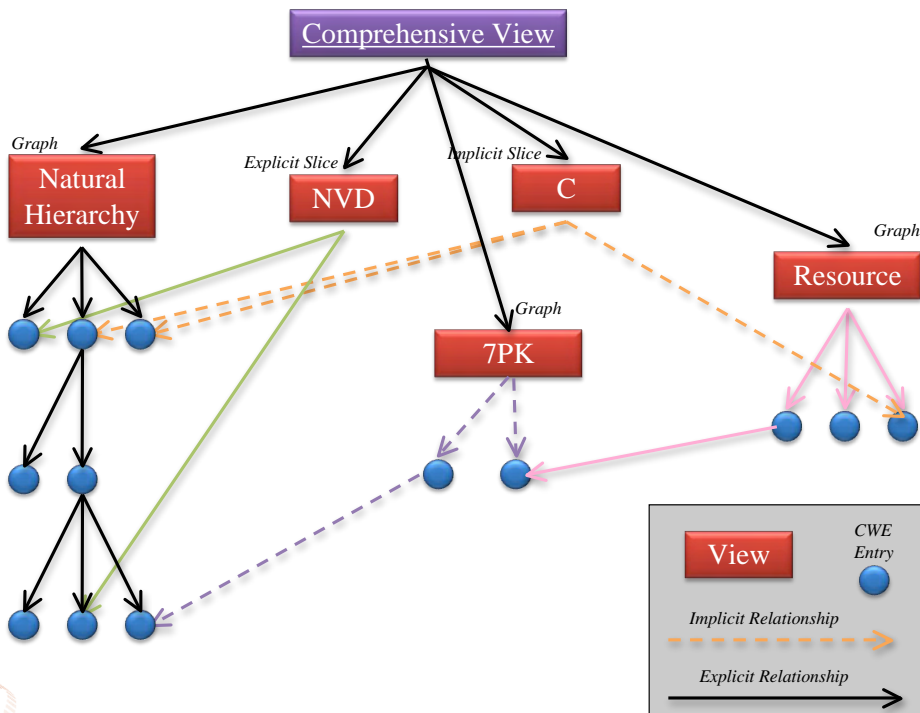


© 2008 MITRE

Node Types: Example



© 2008 MITRE



© 2008 MITRE

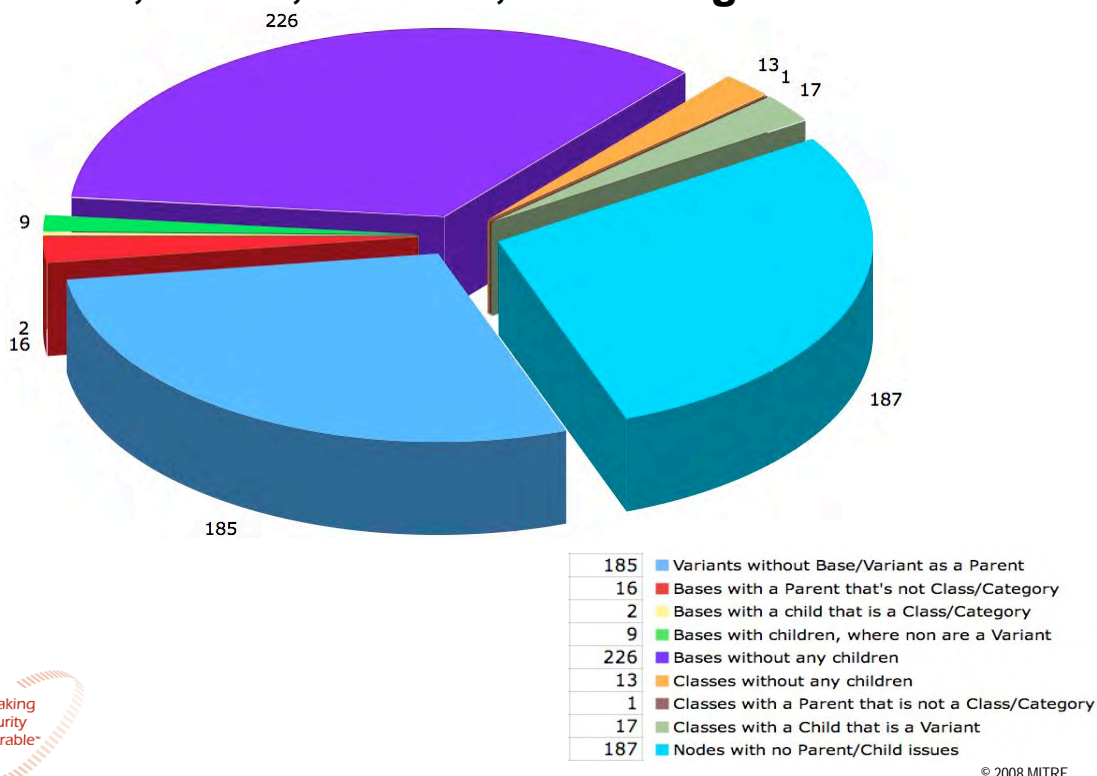
Problems With Categorization

- 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?**



© 2008 MITRE

Role Analysis: Classes, Bases, Variants, and Categories



© 2008 MITRE

Public Difference Reports for Content Changes

Important Changes

A node change is labeled "important" if it is a major field change and the field is critical to the meaning of the node. The critical fields are description, name, and relationships.

Key
D Description
N Name
R Relationships

- R 1 Location
- R 2 Environment
- N R 5 J2EE Misconfiguration: Data Transmission Without Encryption
- R 6 J2EE Misconfiguration: Insufficient Session-ID Length
- R 8 J2EE Misconfiguration: Entity Bean Declared Remote
- N R 9 J2EE Misconfiguration: Weak Access Permissions for EJB Methods
- R 12 ASP.NET Misconfiguration: Missing Custom Error Handling
- N R 14 Compiler Removal of Code to Clear Buffers
- N R 15 External Control of System or Configuration Setting
- R 18 Source Code
- D R 20 Insufficient Input Validation
- D R 22 Path Traversal
- D R 23 Relative Path Traversal
- N 24 Path Traversal: '..\filedir'
- N 25 Path Traversal: '/../filedir'
- N 26 Path Traversal: '/dir/./filename'
- N 27 Path Traversal: 'dir/././filename'
- N 28 Path Traversal: '..\filename'
- N 29 Path Traversal: '\.\filename'
- N 30 Path Traversal: '\\dir.\filename'
- N 31 Path Traversal: 'dir \.\filename'
- N 32 Path Traversal: '...' (Triple Dot)
- N 33 Path Traversal: '...' (Multiple Dot)
- N 34 Path Traversal: '.../..'
- N 35 Path Traversal: '..././'
- D 36 Absolute Path Traversal
- N 37 Path Traversal: '/absolute/pathname/here'
- N 38 Path Traversal: '\absolute\pathname\here'
- N 39 Path Traversal: 'C:dirname'
- N 40 Path Traversal: '\\UNC\share\name\' (Windows UNC Share)

Making Security Measurable™

CWE - Differences between Draft 8 and Draft 9

Summary

Total New	39
Total deprecated	1
Total shared	656
Total important changes	429
Total major changes	463
Total minor changes	399
Total minor changes (no major)	334
Total unchanged	59

Field Change Summary

Affected_Resource	Field	Major	Minor
Alternate_Terms		1	0
Applicable_Platforms		2	0
Back_Rev_Definition		0	0
CVEs_Mentioned		3	0
Class_Names		0	0
Common_Consequences		3	0
Common_Methods_of_Exploitation		0	0
Content_Appear		22	2
Demonstrative_Example		3	2
Description		186	50
Defection_Factor		1	0
Enabling_Factors_for_Exploitation		0	0
Functional_Area		0	0
Likelihood_of_Exploit		0	0
Name		248	1
Note_Relationship		202	71
Observed_Examples		11	1
Parent_Weaknesses		10	0
References		3	0
Related_Attack_Patterns		0	0
Relevant_Properties		3	0
Research_Cases		2	0
Source_Technique		2	0
Time_of_Introduction		39	0
Type		24	377
Weakness_Orality		2	0
White_Box_Definition		0	0

Relationship Changes

Relationship	Total	Draft 8 Tot	Draft 9 Tot	Unchanged	Added to Draft 9	Removed from Draft 9
All	2266	1991	2191	1916	275	75
CanAffect	48	48	47	47	1	1
CanFix	33	33	32	32	1	0
CanPrecede	36	33	35	32	3	1
ChildOf	690	626	658	796	164	32
InRequency	27	25	27	25	2	0
ParentOf	822	826	890	794	96	32
PeerOf	185	175	177	167	10	8
Requires	25	25	25	25	0	0

Node Type Changes

From	To	Total
Unchanged		632
Category	View	5
Category	WeaknessClass	8
WeaknessBase	Composite	5
WeaknessBase	Deprecated	1
WeaknessBase	WeaknessClass	1
WeaknessVariant	Composite	3
WeaknessVariant	WeaknessBase	1

CWE Compatibility & Effectiveness Program (launched Feb 2007)

CWE - CWE Compatibility

CWE Compatibility

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

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

- CWE Searchable** — users may search security elements using CWE identifiers
- CWE Output** — security elements presented to users includes, or allows users to obtain, associated CWE identifiers
- Mapping Accuracy** — security elements accurately link to the appropriate CWE identifiers
- CWE Documentation** — capability's documentation describes CWE compatibility, and how CWE-related functionality in the capability is used
- CWE Coverage** — for CWE-Effectiveness, capability's documentation explicitly lists the CWE identifiers that the capability is effective at locating in

Organizations Participating

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

Products are listed alphabetically by organization name:

cwe.mitre.org/compatible/

TOTALS
 Organizations Participating: 15
 Products & Services: 25

Updated: December 29, 2006

Current Community Contributing to the Common Weakness Enumeration

- AppSIC
- Apple
- Aspect Security
- Booz Allen Hamilton Inc.
- Cenxic
- 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.



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

© 2008 MITRE

A screenshot of a web browser displaying the "CWE - Documents" page. The browser's address bar shows "http://cwe.mitre.org/about/documents.html". The page features the CWE logo and the title "Common Weakness Enumeration - A community-developed dictionary of common software weaknesses". A navigation menu includes "Home", "About CWE", and "Documents". The main content area is titled "Documents" and contains two sections: "Introduction to Vulnerability Theory" and "Structured CWE Descriptions". Each section includes a brief description, a date (July 10, 2007), and author information. Links for "HTML" and "PDF" versions are provided for each document. A sidebar on the left contains a "CWE List" menu with options like "Full Dictionary View" and "Classification Tree". A "Section Contents" sidebar on the right lists "About CWE", "Process", "Sources", "Documents", "Privacy Policy", and "Terms of Use".



© 2008 MITRE

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™

© 2008 MITRE

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™

© 2008 MITRE

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



© 2008 MITRE

Behaviors, Properties, and Manipulations

A **PRODUCT** implements **FEATURES** by performing certain **BEHAVIORS** that operate on **RESOURCES**. The behaviors can be "atomic" or "functional" depending on the level of abstraction being used at the time. Behaviors can be used at the time of a request, to a different time, or to a different socket, memory, or file. Behaviors can be cookies, headers, or other data.

The product's behavior is defined by preserving encoding manipulations. Behaviors are handled as ASCII characters, but they are "equivalent" to the original if the meaning is preserved.

The developer should use appropriate behaviors to implement an **IMPLEMENTATION**. The intended policies and the product's policies are the same.

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 reachability manipulation involves providing a login name first.

TRIGGER requests

FACILITATION - limitation on alphanumeric characters, Perl/PHP, arbitrary manipulation already

Manipulation composition

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.

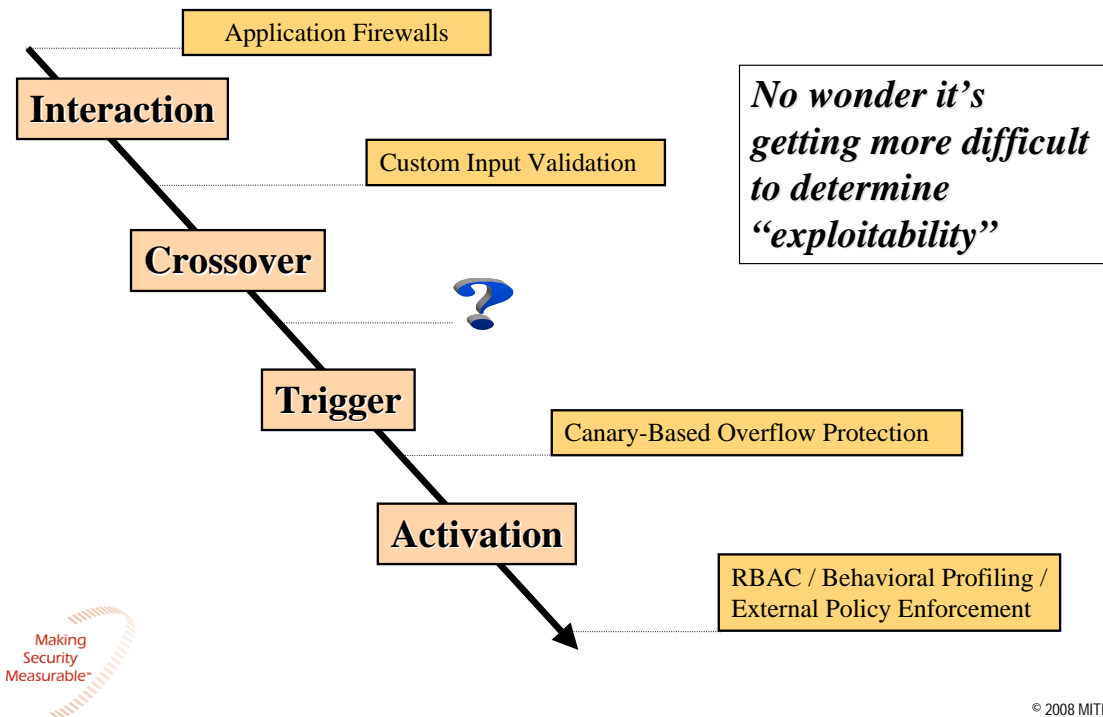
ACTIVATION POINT - the location where the attacker's "payload" becomes activated; presumably the payload involves the incorrect behaviors.

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.



© 2008 MITRE

Artifact Labels and Protection Mechanisms



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:

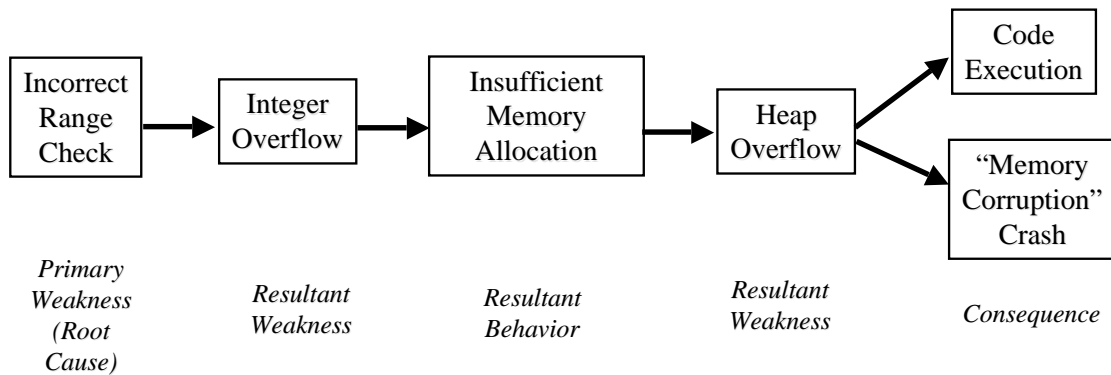
Interaction: 3
Crossover: between 4 and 5
Trigger: 5
Activation: 5 (if DoS intended), outside code (if code execution)

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

Many Vulnerabilities are Multi-Factor

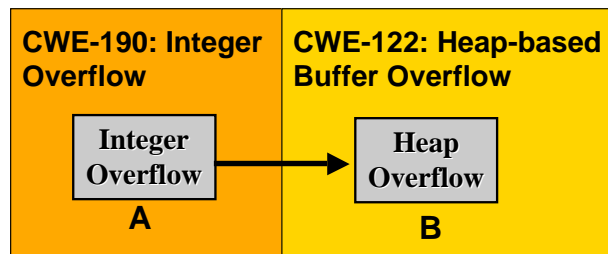


One of the main problems with classification and terminology is that ANY behavior in the chain could be regarded as the vulnerability.



© 2008 MITRE

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.



© 2008 MITRE

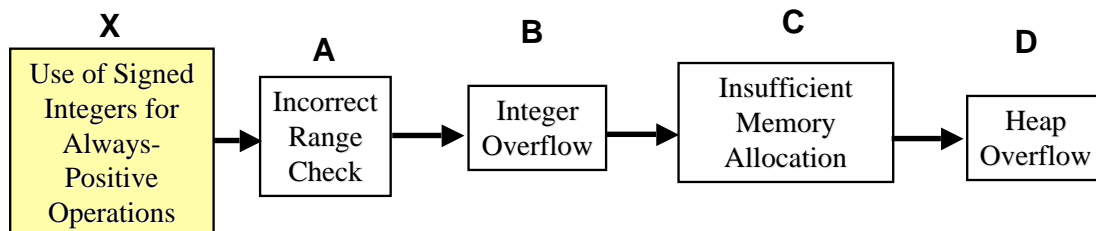
Protection Schemes

- 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



© 2008 MITRE

Chain Example: Failed Protection Mechanism with Resultant Issues



height = -65534; width = -65534

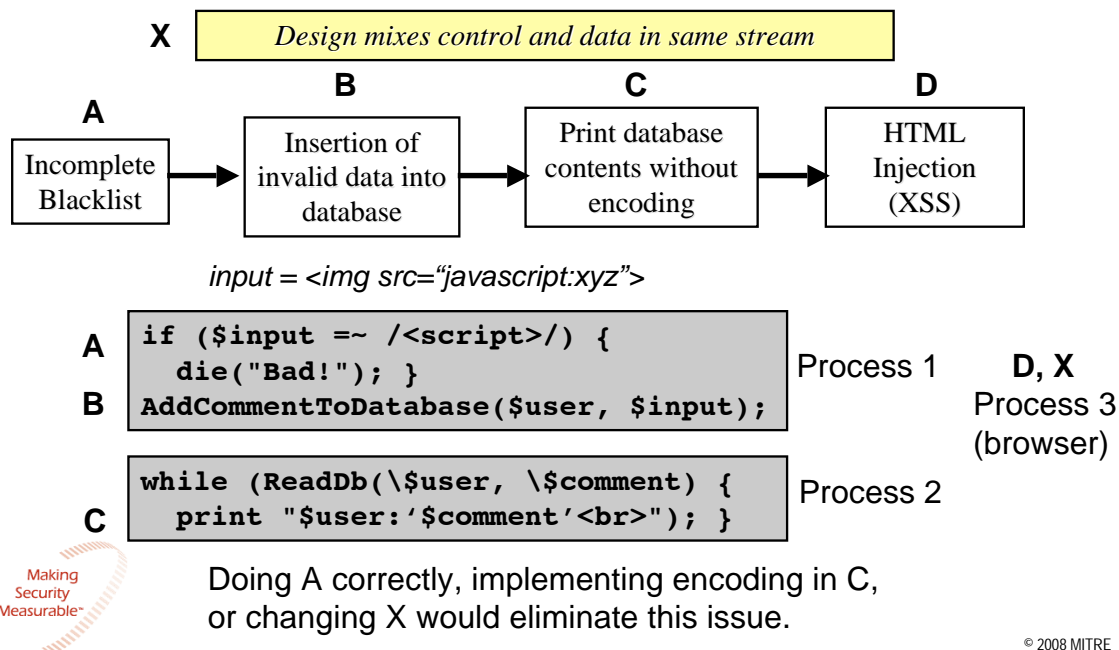
Assumption: the range check will prevent an overflow from occurring.

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



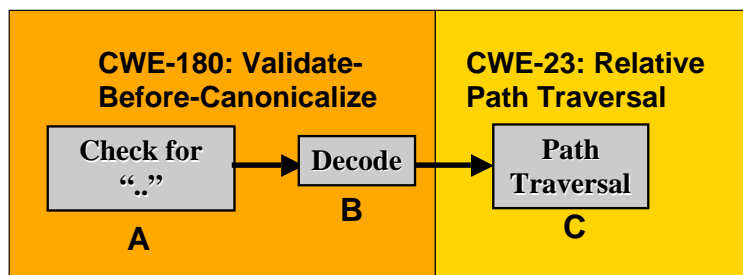
© 2008 MITRE

Failed Protection Mechanism with Resultant Issues (XSS)



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.



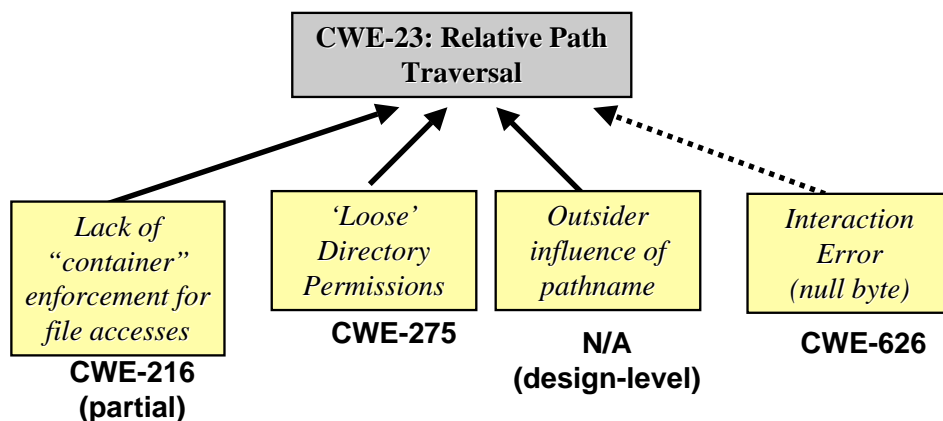
Null byte allows access of files not ending in .txt

```

lang = %2e./%2e./%2e/etc/passwd%00
$home = "/home/www/118n/";
$input = WebParam('lang');
A if ($input =~ /\.\.\/) {
      die("Bad!"); }
B $input = URLdecode($input);
      $fname = $home + "$input" + ".txt";
C ReadFileAndDumpToUser($fname);
  
```



Composite: Path Traversal

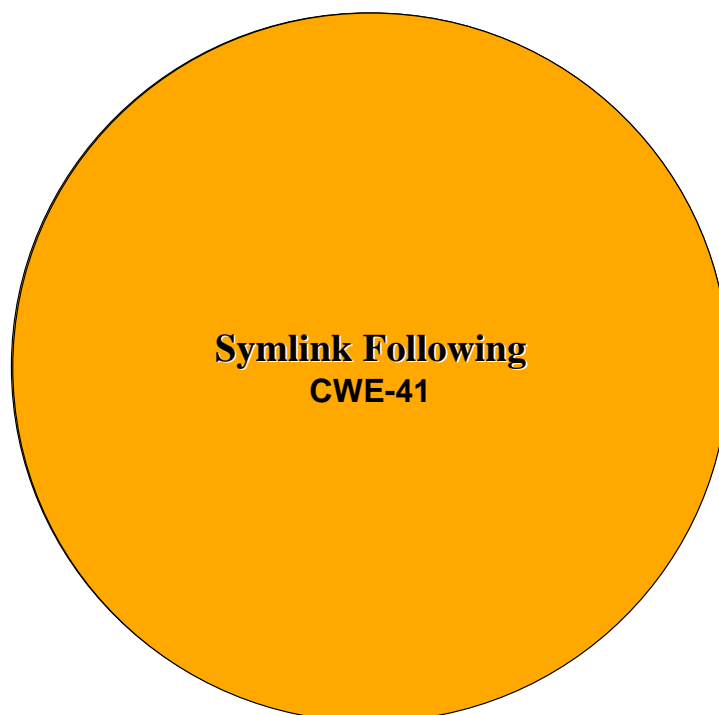


- Application can potentially access anywhere user 'nobody' can
- No real built-in OS permissions for 'cannot navigate above this directory'
- Null bytes widen the scope – 'cut off' .txt extension
- Influence of pathname is typically a design-level decision, and can be done safely with proper pathname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



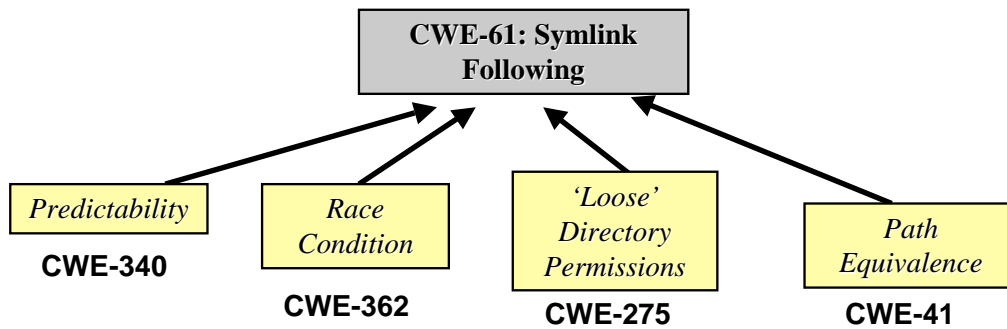
© 2008 MITRE

Symbolic Link Following (composition)



© 2008 MITRE

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



© 2008 MITRE

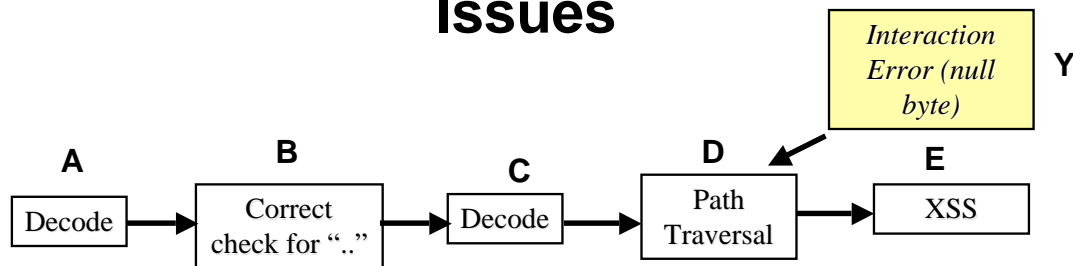
Integer Overflow Discussion

- 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



© 2008 MITRE

Incorrect Ordering with Resultant Issues



X
Lack of "container" enforcement for file accesses

Null byte allows access of files beyond .txt



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

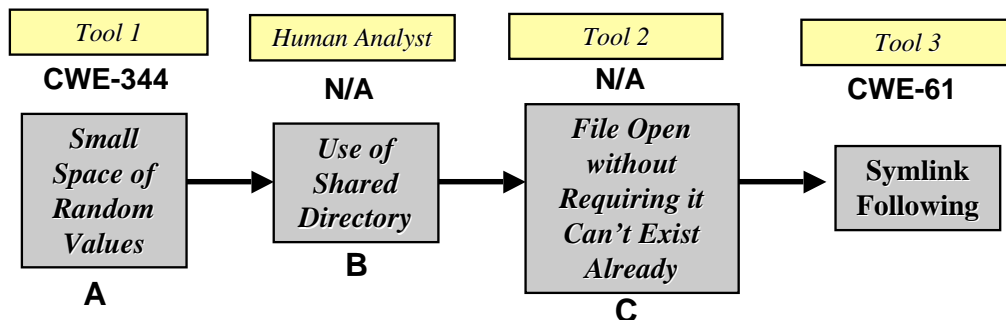
```

A $home = "/home/www/118n/";
B $input = urldecode(WebParam('lang'));
    if (($input =~ /^\/\//) ||
        ($input =~ /\.\.\/)) {
C     die("Bad!"); }
    $input = urldecode($input);
    $fname = $home + "$input" + ".txt";
D, E DumpFileOutput($fname);
  
```

© 2008 MITRE

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?



© 2008 MITRE

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



© 2008 MITRE

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



© 2008 MITRE

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='Cause') or (/Relationship.Name='1091') = (/Relationship.Target_ID = /Relationship.Nature='Cause')

CWEs in this view	Total CWEs
Total	43 out of 993
Views	0 out of 14
Categories	1 out of 54
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') - (601)
- Failure to Resolve Case Sensitivity - (176)
- 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 - (239)
- Heap-based Buffer Overflow - (122)
- Improper Null Termination - (170)
- Incomplete Backlist - (184)
- Incomplete Backlist to Cross-Site Scripting - (692)
- Incorrect Output Sanitization for Logs - (117)
- Insufficient Control of File Name for Include/Require Statement in PHP Program (aka 'PHP File Inclusion') - (1)
- 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)
- Path Equivalence: 'filename' (Trailing Space) - (46)
- Path Equivalence: 'filename' (Trailing Space) - (46)
- Reachable Assertion - (617)
- Signed to Unsigned Conversion Error - (195)
- Unbounded Transfer ('Classic Buffer Overflow') - (120)
- Unbounded 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 - (327)
- 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 Backlist to Cross-Site Scripting

Chains

- CWE-16 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-32 Failure to Sanitize CRLF Sequences (aka 'CRLF Injection')
- 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 Backlist to Cross-Site Scripting
- CWE-171 Cleansing, Canonicalization, and Comparison Errors
- CWE-289 Authentication Bypass by Alternate Name
- CWE-123 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 Backlist
- 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-28 Insufficient Control of File Name 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-122 Heap-based Buffer Overflow
 - 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 Write-what-where Condition
- 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-122 Write-what-where Condition
- 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 File Name 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-239 Hard-Coded Password
- CWE-321 Use of Hard-coded Cryptographic Key
- CWE-472 External Control of Assumed-Immutable Web Parameter

Software Assurance
Community Resources and Information Clearinghouse

Sponsored by DHS National Cyber Security Division

What is Software Assurance?

Software Assurance (SWA) is the level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at any time during its life cycle, and that the software functions in the intended manner (from CISA 4009 IA Glossary - see Wikipedia for definitions and descriptions).

As part of the DHS risk mitigation effort, the Software Assurance Program seeks to reduce software vulnerabilities, minimize exploitation, and address steps to improve the routine development of trustworthy software products with predictable execution, and to improve diagnostic capabilities to analyze systems for hidden vulnerabilities.

The SWA framework encourages the production, evaluation, and acquisition of better quality and more secure software, providing focuses in these four areas:

- People: Education and training for developers and users
- Process: Sound practices, standards, and practical guides for the development of secure software
- Technology: Diagnostic tools, cyber security R&D and measurement
- Acquisition: Specifications and guidelines for acquisition and outsourcing

The Software Assurance Forum and several working groups composed of volunteers from government, industry, and academia are helping the Software Assurance Program achieve its objectives in these focus areas.

Software Assurance Forum

The Software Assurance Forum has provided a collaborative venue for stakeholders to share and advance techniques and technologies relevant to software security. Software Assurance: A State of the Art Report (SOAR) represents an output of collaborative efforts of organizations and individuals in the SWA Forum and working groups. The SOAR provides an overview of the current state of the environment in which software must operate and surveys current and emerging activities and organizations involved in promoting various aspects of software security assurance. The report also describes the variety of techniques and technologies in use in government, industry, and academia for specifying, acquiring, producing, assessing, and deploying software that can, with a justifiable degree of confidence, be said to be secure. The report also presents observations about noteworthy trends in software security assurance as a discipline. Many other SWA resources are provided by the SWA working groups.

Why is Software Assurance Critical?

The nation's critical infrastructure (energy, transportation, telecommunications, etc.), businesses, and services are extensively and increasingly controlled and enabled by software. Vulnerabilities in that software put those resources at risk. The risk is compounded by software size and complexity, the use of software produced by unvetted suppliers, and the interdependence of software systems. Software assurance deals with root of the problem by improving software security.

How is Software Assurance Advanced?

The Software Assurance Forum has provided a collaborative venue for stakeholders to share and advance techniques and technologies relevant to software security. Software Assurance: A State of the Art Report (SOAR) represents an output of collaborative efforts of organizations and individuals in the SWA Forum and working groups. The SOAR provides an overview of the current state of the environment in which software must operate and surveys current and emerging activities and organizations involved in promoting various aspects of software security assurance. The report also describes the variety of techniques and technologies in use in government, industry, and academia for specifying, acquiring, producing, assessing, and deploying software that can, with a justifiable degree of confidence, be said to be secure. The report also presents observations about noteworthy trends in software security assurance as a discipline. Many other SWA resources are provided by the SWA working groups.

Build Security In
Setting a Higher Standard for Software Assurance

Home | People | Process | Technology | Acquisition | Working Groups

DHS SWA WG Output

These are resources that are either part of other DHS-sponsored work or closely related thereto.

Supplementary Department of Homeland Security Resources (Ordered by Date)

- Software Assurance Curriculum Organization (10/30/07)**
The Department of Homeland Security (DHS) Software Assurance Program is seeking review and comment on Software Assurance: A Curriculum Guide to the Common Body of Knowledge for Producers, Acquirers, and System Secure Software (version 1.2) and the associated SWA Clear Principles Matrix, the Clear Principles SOAR Matrix and Principles Organization. Towards an Organization for Software System Security Principles and Guidelines (version 2.06) are also available for reference and review. Please see the comment form for them and submit comments by December 4, 2007 to Krissy Heister and Samuel Bedard.
- IT Security Essential Body of Knowledge (10/25/07)**
The IT Security Essential Body of Knowledge (EBK) further clarifies key IT security terms and concepts for well-defined competencies, identifies national security roles, defines the primary functional perspectives, and establishes an IT Security Role, Competency, and Functional Matrix. We seek continued input from the community to ensure that it accurately reflects baseline skill requirements of the nation's IT security workforce.
- Software Assurance (SWA) in Acquisition: Missing Links to SWA Integration (10/20/07)**
This guide provides information on incorporating software assurance throughout the acquisition process from the acquisition planning phase to contracting, implementation and acceptance, and follow-on phases. For each phase, the guide covers software assurance concepts, recommended strategies, and acquisition management tips. The guide also includes recommended Request for Proposal (RFP) and contract language and due diligence questionnaires that may be tailored by acquisition officials to facilitate the contract evaluation process.
- Software Assurance (SWA) in Acquisition: Missing Links to SWA Integration (10/20/07)**
The purpose of this report is to collect and present information on how the need for software assurance affects software project management. The impact of software assurance on the tasks and concerns of a project manager are addressed, first in terms of diagnosis, and then in terms of the life cycle. A second goal of this SOAR is to provide tools and resources for quantifying the effects of software assurance on software development, both in terms of planning (cost estimation/budgeting) and in terms of overall cost-effectiveness and return on investment.
- SOAR on Software Security Assurance (08/07)**
The Software Assurance Forum has provided a collaborative venue for stakeholders to share and advance techniques and technologies relevant to software security. The report of the Software Assurance Forum (SOAR) represents an output of collaborative efforts of organizations and individuals in the SWA Forum and working groups. The SOAR provides an overview of the current state of the environment in which software must operate and surveys current and emerging activities and organizations involved in promoting various aspects of software security assurance. The report also describes the variety of techniques and technologies in use in government, industry, and academia for specifying, acquiring, producing, assessing, and deploying software that can, with a justifiable degree of confidence, be said to be secure. The report also presents observations about noteworthy trends in software security assurance as a discipline. Many other SWA resources are provided by the SWA working groups.

Software Assurance

Home | People | Process | Technology | Acquisition | Working Groups

Focus Area: Technology

In this section you'll find resources relating to software security measurement, SWA research and development, and SWA testing and diagnostic tools.

- NIST SAMATE Reference Dataset: Software development artifacts with known security weaknesses
- Common Weakness Enumeration: A dictionary of common software weaknesses
- Common Attack Patterns Enumeration and Classification: A taxonomy of attack patterns
- Security Measurement: A white paper published by Practical Software and Systems Measurement
- Federal Plan for Cyber Security and Information Assurance Research and Development: Available for download on the National Coordination Office for Networking and Information Technology Research and Development site.

The Data & Analysis Center for Software

RELATED WORKING GROUPS

Efforts of these working groups fall within this focus area:

- Technology, Tools and Product Evaluation Working Group
- Measurement Working Group
- Malware Working Group

Making Security Measurable

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

Automation of vulnerability management, security measurement, and compliance (e.g. FISMA).

Resource Status

NVD contains: 26736 CVE Vulnerabilities

- 114 Checklists
- 91 US-CERT Alerts
- 1997 US-CERT Vuln Notes
- 2966 OVAL Queries
- 12410 Vulnerable Products

Last updated: 09/26/07

CVE Publication rate: 16 vulnerabilities / day

Email List

Select the email list(s) you wish to join, enter your e-mail address and press "Add" to receive NVD announcements or SCAP information.

- NVD Announcements
- SCAP Announcements
- SCAP Discussion List
- XCCDF Discussion List

Workload Index

Vulnerability Workload Index: 9.06

About Us

NVD is a product of the NIST Computer Security Division and is sponsored by the Department of Homeland Security's National Cyber Security Division. It supports the

Overview

SQL injection vulnerability in mods/banners/navist.php in Clansphere 2007.4 allows remote attackers to execute arbitrary SQL commands via the cat_id parameter to index.php in a banners action.

Impact

CVSS Severity (version 2.0): CVSS v2 Base score: 7.5 (High) (AV:N/AC:L/Au:N/C:P/I:P/A:P) (legend)

Impact Subscore: 6.4

Exploitability Subscore: 10.0

Access Vector: Network exploitable

Access Complexity: Low

Authentication: Not required to exploit

Impact Type: Provides unauthorized access, Allows partial confidentiality, integrity, and availability violation, Allows unauthorized disclosure of information, Allows disruption of service

References to Advisories, Solutions, and Tools

External Source: BID (disclaimer)

Name: 25770

Hyperlink: <http://www.securityfocus.com/bid/25770>

External Source: MILWORM (disclaimer)

Name: 4443

Hyperlink: <http://www.milw0rm.com/exploits/4443>

Vulnerable software and versions

Configuration 1

- Clansphere, Clansphere, 2007.4

Technical Details

Vulnerability Type (View All)
SQL Injection (CWE-89)

CVE Standard Vulnerability Entry:
<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-5061>

Common Platform Enumeration:

NVD XML feeds also include CWE

Vulnerability Type (View All)
SQL Injection (CWE-89)

CWE Common Weakness Enumeration
A Consistently-Developed Dictionary of Software Weakness Types

Version: 2.093 (Last 3 CWEs are Individual Dictionary Definitions (Draft 9))

CWE-89 Individual Dictionary Definition (Draft 9)

Weakness ID: 89 (Insecure) **Basic Information**

Failure to Sanitize Data into SQL Queries (aka "SQL Injection")

Description

The application fails to adequately filter SQL queries from user-controllable input. This can lead to such input being incorporated as SQL, rather than ordinary user data and is the critical part of a database-generated SQL query. This is a specific form of an injection problem, one that exploits effects SQL databases, in which SQL commands are injected into data streams in order to affect the execution of arbitrary database SQL statements.

Likelihood of Exploit

Very High

Common Consequences

Confidentiality: Since SQL databases generally hold sensitive data, loss of confidentiality is a frequent problem with SQL injection vulnerabilities. Authorization: If user SQL commands are used to check user names and passwords, it may be possible to connect to a system as another user with no previous knowledge of the password. Integrity: If authorization information is held in a SQL database, it may be possible to change this information through the successful exploitation of a SQL injection vulnerability. Impact: Just as it may be possible to read sensitive information, it is also possible to make changes or even delete this information with a SQL injection attack. Requirements specification: A non-SQL style database which is not subject to the flaw may be chosen. Design: Follow the principle of least privilege when creating user accounts to a SQL database. Users should only have the minimum privileges necessary to use their account. If the requirements of the system indicate that a user can read and modify their own data, then limit their privileges so they cannot read/write other's data. Design: Duplicate any filtering done on the client side on the server side. Implementation: Implement SQL string use prepared statements that bind variables. Prepared statements that do not bind variables can be vulnerable to attack.

Potential Mitigations

© 2008 MITRE

Source: <http://www.securityfocus.com/bid/25770>

Making Security Measurable

SANS & Partners Announce First Secure Coding Assessment and Certification Exams for Programmers

GAC Secure Software Programmer (SSSP) Certification Exam

SAMATE Reference Dataset

NIST SAMATE Reference Dataset Project

NIST Draft Special Publication 500-268

Source Code Security Analysis Tool Functional Specification Version 1.0

Software Assurance

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

- Establish a common framework of software properties that can be used to represent any/all classes of software so software suppliers and acquirers can represent their claims and arguments/respective views, along with the corresponding evidence, employing automated tools to address scores.
- Verify that products have sufficiently analyzed these characteristics in advance of product acquisition, so that system engineers/integrators can use these products to build (compose) larger assured systems with them.
- Enable industry to improve visibility into the current status of software assurance during development of its software.

Contact: Co-Chairs: Ms. Dorena Camara, KDM Analytics; Mr. J.D. Baker, SANS Systems.

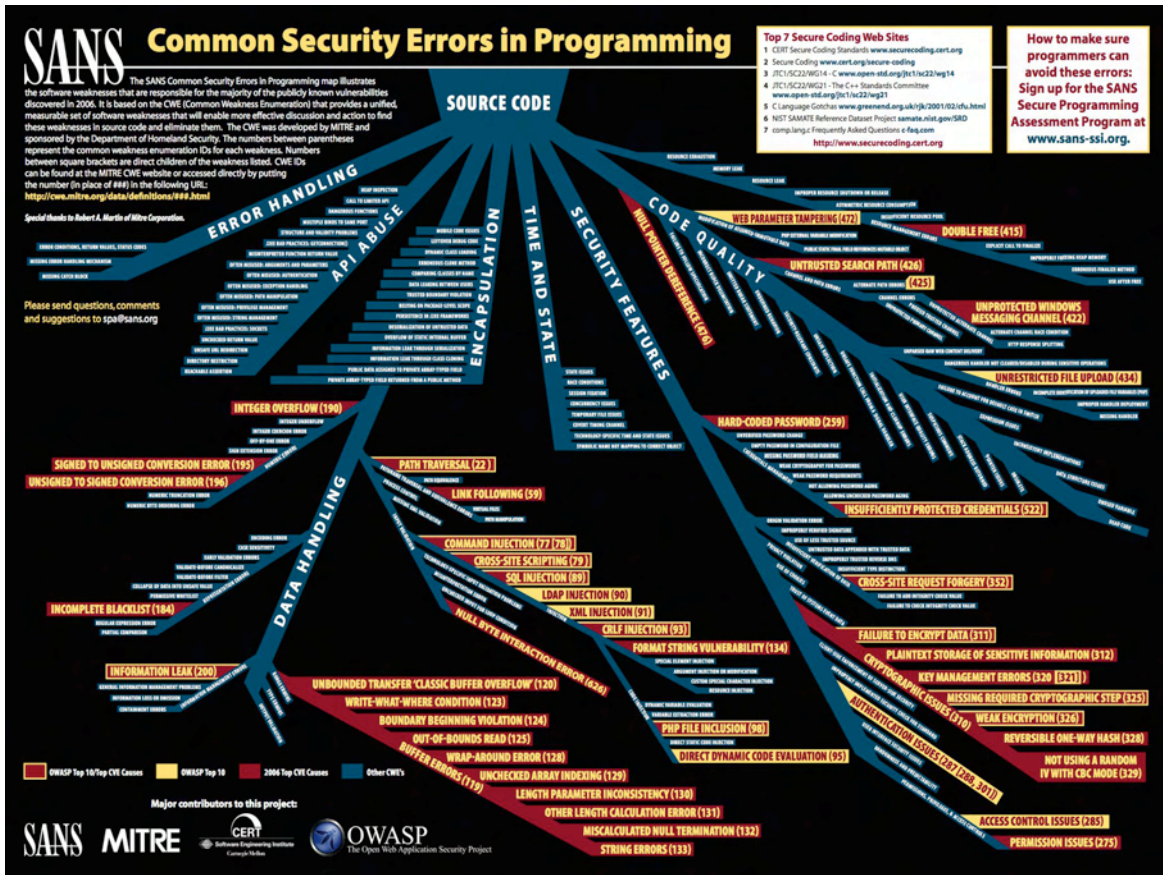
How To Get Involved: If you are interested in getting involved with this group, want more information or would like to come as a guest to an upcoming meeting and obtain temporary access to the mailing list, please contact one of our Account Representatives or contact one of the Chairs.

Software Assurance Information Day: SANS Technical Meeting Special Event

Technology Laboratory (ITL), Software and Conformance Testing Division

Michael Kass
Michael Koo

© 2008 MITRE



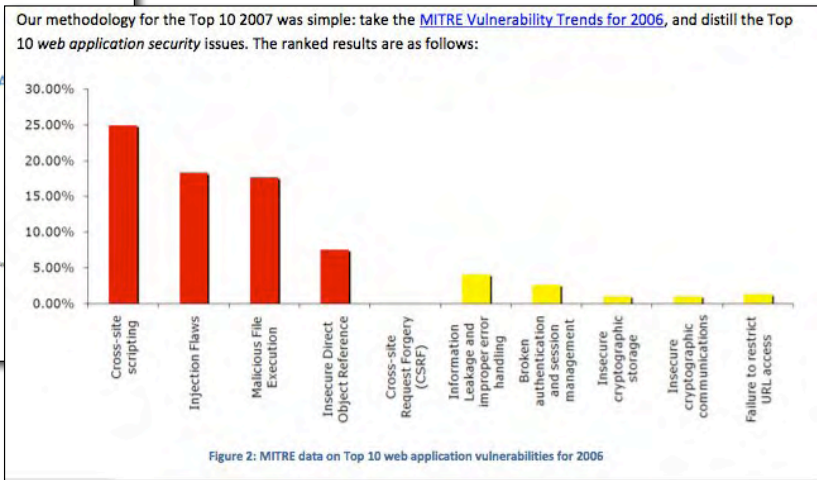
OWASP Top Ten 2007

OWASP TOP 10

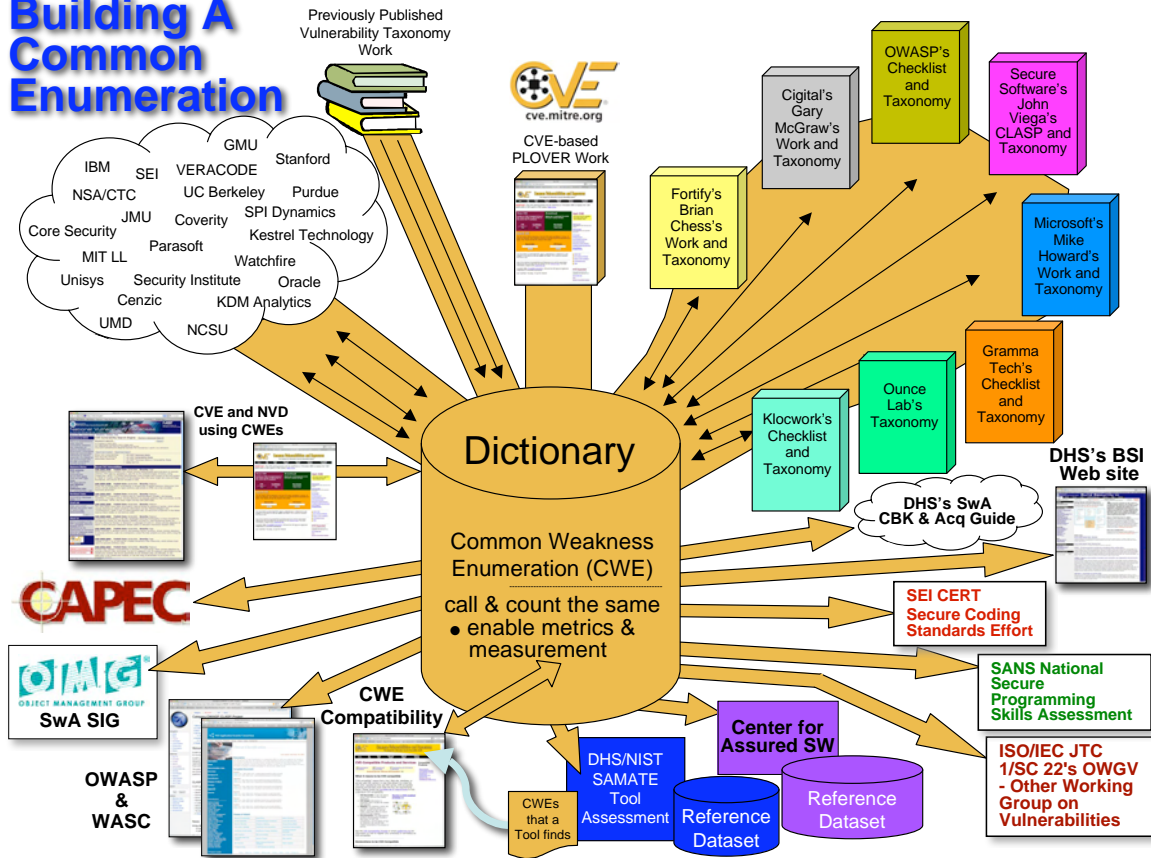
THE TEN MOST CRITICAL WEB APPLICATION SECURITY VULNERABILITIES

2007 UPDATE

© 2002-2007 OWASP Foundation
This document is licensed under the Creative Commons Attribution-ShareAlike 2.5 license



Building A Common Enumeration



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



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



© 2008 MITRE

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



© 2008 MITRE

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



© 2008 MITRE

Common Attack Pattern Enumeration and Classification (CAPEC)

The screenshot displays the CAPEC website interface for the entry '31: Accessing/Intercepting/Modifying HTTP Cookies'. The page is divided into several sections:

- Header:** CAPEC - Individual CAPEC Dictionary Definition (Release 1.1)
- Navigation:** Home > CAPEC List > Individual CAPEC Dictionary Definition (Release 1.1)
- Left Sidebar:**
 - CAPEC List:** Full CAPEC Dictionary, Classification Tree, Other Views
 - About CAPEC:** Documents, Resources
 - Community:** Related Activities, Collaboration List
 - Contact Us:** Search the Site
- Main Content Area:**
 - Individual CAPEC Dictionary Definition (Release 1.1)**
 - Attack Pattern ID:** 31
 - Typical Likelihood of Exploit:** High
 - Description:** This attack relies on the use of HTTP Cookies to store credentials, state information... The first form of this attack involves accessing HTTP Cookies to mine for potential... The second form of this attack involves intercepting this data as it is transmitted... The third form is when the cookie's content is modified by the attacker before it convinces the target server to operate on the falsified information.
 - Attack Execution Flow:**
 - Exploit:** 1. Obtain copy of cookie: The attacker first needs to obtain a copy of the cookie. The attacker could be somebody sniffing on a network to get a copy of HTTP cookies.
 - Attack Step Techniques:**
 - Description:** Obtain cookie from local filesystem by C:\Documents and Settings*\Cookies and C:\Documents and Settings*\Internet Explorer*\Cookies folders (see Windows).
 - Self cookie using a network sniffer such as Wireshark
 - Obtain cookie from local memory or Register using a library such as the Firefox Cookie Manager or IECC
 - Steal cookie via a cross-site scripting attack.
 - Guess cookie contents if it contains predictable information.
 - Indicators of Susceptibility:**
 - ID:** Type Description
 - 01511: Positive Cookies used in web application.
 - 01512: Negative Cookies not used in web application.
 - Solutions:**
 - ID:** Type Description
 - 01513: Access Cookie captured by attacker.
 - 01514: Failure Cookie cannot be captured by attacker.
 - Security Control:**
 - ID:** Type Description
 - 01515: Preventative To prevent network sniffing, cookies should be transmitted over HTTPS and not plain HTTP. To enforce this on the client side, the set on cookies (java.cookie.Cookie.setSecure() in Java, secure flag in setcookie() function in PHP, etc.).
- Right Sidebar:**
 - Attack Prerequisites:** Target server software must be a HTTP daemon that relies on cookies.
 - Typical Likelihood of Exploit:** High
 - Methods of Attack:**
 - Modification of Resources
 - API Abuse
 - Protocol Manipulation
 - Time and State
 - Examples-Instances:**
 - Description:** There are two main attack vectors for exploiting poorly protected session variables like cookies. One is the local machine itself which can be exploited directly at the physical level or indirectly through XSS and phishing. In addition, the man in the middle attack relies on a network sniffer, proxy, or other intermediary to intercept the subject's credentials and use them to impersonate the digital subject on the host. The issue is that once the credentials are intercepted, impersonation is trivial for the attacker to accomplish if no other protection mechanisms are in place.
 - Attacker Skill or Knowledge Required:** Low -> To overwrite session cookie data, and submit targeted attacks via HTTP High -> Exploiting a remote buffer overflow generated by attack
 - Injection Vector:** HTTP cookie
 - Payload:** Malicious input delivered through cookie in HTTP Request.
 - Activation Zone:** Client software, such as a browser and its component libraries, or an intermediary
 - Payload Activation Impact:** 1. Enables attacker to leverage state stored in cookie 2. Enables attacker a vector to attack web server and platform
 - Related Weaknesses:**

CWE-ID	Weakness Name	Weakness Relationship Type
353		Targeted
352		Targeted
313		Targeted
522		Targeted
20		Targeted
313		Targeted
304		Targeted
472		Secondary
 - Context Description:**

Purpose	CIA Impact	Confidentiality Impact	Integrity Impact	Availability Impact
Exploitation	High	High	High	Low
 - Technical Context:**

Architectural Paradigm	Framework	Platform	Language
Client-Server	All	All	All
 - References:** G. Hoglund and G. McGraw. Exploiting Software: How to Break Code. Addison-Wesley, February 2004.
 - Source:**

Submitter	Organization	Date	Comment
G. Hoglund and G. McGraw	Exploiting Software: How to Break Code. Addison-Wesley, February 2004.		
Digital, Inc.		2007-01-01	
 - Modification(s):**

Modifier	Organization	Date	Comment
Gunnar Peterson	Digital, Inc.	2007-02-28	Filtered out content to CAPEC schema from the original descriptions in "Exploiting Software"
Sean Barnum	Digital, Inc.	2007-03-09	Review and revise
Richard Struse	VORNE, Inc.	2007-03-26	Review and feedback leading to changes in Name and Description
Sean Barnum	Digital, Inc.	2007-04-13	Modified pattern content according to review and feedback
Amit Sathya	Digital, Inc.	2007-10-29	Added extended Attack Execution Flow



How do you identify relevant attack patterns?

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



© 2008 MITRE

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



© 2008 MITRE

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



© 2008 MITRE

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



© 2008 MITRE

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))



© 2008 MITRE

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



© 2008 MITRE



[makingsecuritymeasurable.mitre.org]

Making Security Measurable

A Collection of Information Security Community Standardization Activities and Initiatives

Home | Current Collection | Feedback Requested

Measurable security pertains at a minimum to the following areas:

- Vulnerability Management
- Asset Security Assessment
- Configuration Guidance
- Malware Response
- Threat Analysis
- Intrusion Detection
- Asset Management
- Patch Management
- Incident Management

Enumerations

- CVE** - Common Vulnerabilities and Exposures (CVE®) - common vulnerability identifiers
- CWE** - Common Weakness Enumeration (CWE™) - list of software weakness types
- CAPEC** - Common Attack Pattern Enumeration and Classification (CAPEC™) - list of common attack patterns
- CME** - Common Malware Enumeration (CME™) - common identifiers for viruses, worms, and other malicious code
- CCE** - Common Configuration Enumeration (CCE™) - common security configuration identifiers
- CPE** - Common Platform Enumeration (CPE™) - common platform identifiers
- SANS Top Twenty** - SANS/IFIP consensus list of the Twenty Most Critical Internet Security Vulnerabilities that uses CVE-IDs to identify the issues
- OWASP Top Ten** - ten most critical Web application security flaws
- WASC Web Security Threat Classification** - list of Web security threats

Languages

- OVNL** - Open Vulnerability and Assessment Language (OVALS) - standard for determining vulnerability and configuration issues
- CRF** - Common Result Format (CRF™) - standardized assessment result format for conveying findings based on common names and naming schemes
- CEE** - Common Event Expression (CEE™) - standardizes the way computer events are described, logged, and exchanged
- OVAL Interpreter** - free tool for collecting information for testing, carrying out OVAL Definitions, and presenting results of the tests
- Benchmark Editor™** - free tool that enhances and simplifies creation and editing of benchmark documents written in XCCDF and OVAL
- Extensible Configuration Checklist Description Format (XCCDF)** - specification language for uniform expression of security checklists, benchmarks, and other configuration guidance
- Common Vulnerability Scoring System (CVSS)** - open standard that conveys vulnerability severity and helps determine urgency and priority of response
- Common Announcement Interchange Format (CAIF)** - XML-based format created to store and exchange security announcements in a normalized way
- OMG Semantics of Business Vocabulary and Business Rules (SBVR)** - language for interchange of business vocabularies and rules among organizations and software tools

Repositories

- OVAL Repository** - community-developed OVAL Vulnerability, Compliance, Inventory, and Patch Definitions
- National Vulnerability Database (NVD)** - U.S. vulnerability database based on CVE that integrates all publicly available vulnerability resources and references
- NIST Security Content Automation Protocol (SCAP)** - security content for automating technical control compliance activities, vulnerability checking, and security measurement
- Red Hat Repository** - OVAL Patch Definitions corresponding to Red Hat Errata security advisories
- Center for Internet Security (CIS) Benchmarks** - best-practice security configurations accepted for compliance with FISMA, the ISO standard, GLB, SOX, HIPAA, and FIRPA, and other regulatory requirements for information security
- DISA Security Technical Implementation Guides (STIGS)** - U.S. Defense Information Systems Agency's (DISA) STIGS are configuration standards for DOD information assurance and information assurance-enabled devices and systems

View the current collection of organizations, activities, and initiatives.

Disclaimer

The Web site is hosted by The MITRE Corporation. © 2008 The MITRE Corporation. CVE is a registered trademark and the Making Security Measurable logo, CCE, CME, CWE, CPE, and OVAL are trademarks of The MITRE Corporation. All other trademarks are the property of their respective owners. Contact us: measurablesecurity@mitre.org

Page Last Updated: January 17, 2008

Nabble - CWE Research List forum & mailing list archive

http://www.nabble.com/CWE-Research-List-f26532.html

Nabble | Computer Security » CWE Research List

Parent Categories/Forums: Computer Security : Common Weakness Enumeration

CWE Research List

Search:

This forum is an archive for the mailing list: cwe-research-list@lists.mitre.org (mailing list options). Messages posted here will be sent to this mailing list.

CWE Common Weakness Enumeration
A Community-Developed Dictionary of Software Weakness Types

CWE Research - A lightly moderated public forum to discuss CWE definitions, suggest potential definition expansion(s), and/or submit new definitions. General discussion of the vulnerabilities themselves is also welcome.

Child Forums (0): None

[Post New Message](#) :: [Alert me of new posts](#)

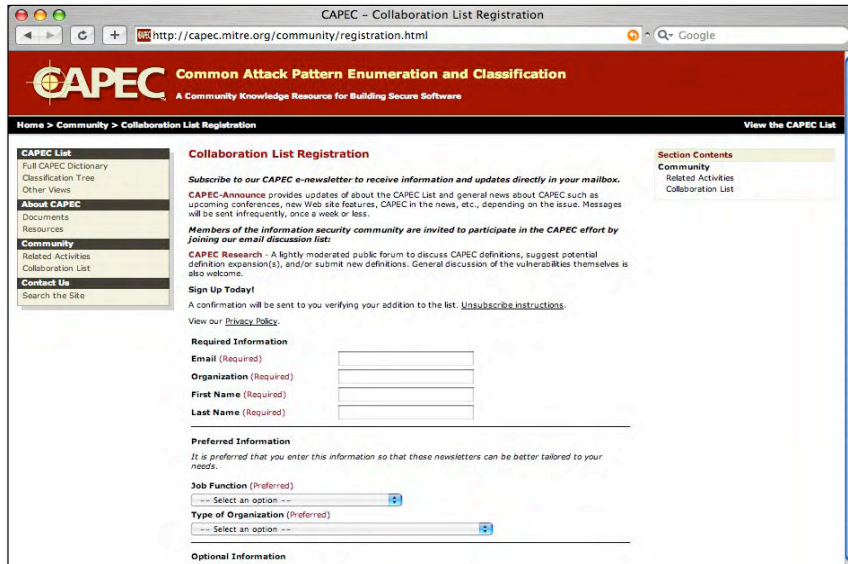
Thread (6 Threads)	Rating	Replies	Last Message
Discussion point: CONSPEC - Context-specific Issues by Steven M. Christey-2	☆☆☆	29	09:32am Chris Wysopal-2
On the viewpoint of CWE descriptions by Nikolai Mansourov	☆☆☆	0	Nov 14 Nikolai Mansourov
Examples of Detailed Changes to Context-Specific CWE Nodes (CONSPEC) by Steven M. Christey-2	☆☆☆	2	Oct 19 Pascal Meunier-3
Managing Node Restructuring in CWE by Steven M. Christey-2	☆☆☆	0	Sep 13 Steven M. Christey-2
New CWE strategy, discussion points, and other documents by Steven M. Christey-2	☆☆☆	0	Sep 13 Steven M. Christey-2
CWE Review Activities in September by Steven M. Christey-2	☆☆☆	0	Sep 10 Steven M. Christey-2

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**





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



© 2008 MITRE

Acronyms

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