Common Weakness Enumeration — CWE™

A Community-Developed Dictionary of Software Weakness Types

CWE, targeted to developers and security practitioners, is a formal list of software weaknesses that:

- Serves as a common language for describing software security weaknesses in architecture, design, or code.
- Serves as a measuring stick for software security tools targeting these weaknesses.
- Provides a common baseline definition for weakness identification, mitigation, and prevention efforts.
- Is industry-endorsed via the CWE Community and CWE-Compatible Products.

Some Common Types of Software Weaknesses:

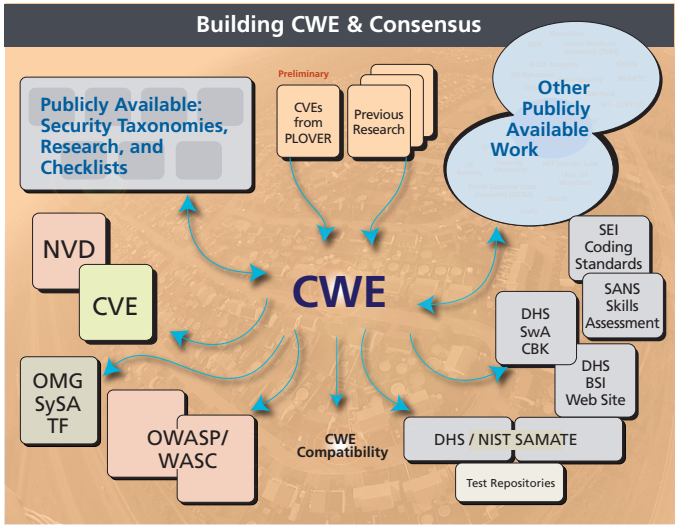
- Buffer overflows, format strings, etc.
- Structure and validity problems
- Common special element manipulations
- Channel and path errors
- Handler errors
- User interface errors
- Pathname traversal and equivalence errors
- Authentication errors
- Resource management errors
- Insufficient verification of data
- Code evaluation and injection
- Randomness and predictability

Challenge

Software acquirers want assurance that the software products they are obtaining are reviewed for known types of exploitable security weaknesses, and the acquisition groups in large government and private organizations are moving forward to use these types of reviews as part of future contracts. Until recently the tools and services that could be used for this type of review were new at best and there were no nomenclature, taxonomies, or standards to define the capabilities and coverage of them. That made it difficult to comparatively decide which tool or service was best suited for a particular job. What was needed was a standard list and classification of software security weaknesses to serve as a unifying language of discourse and a measuring stick for tools and services.

Solution

CWE is a community-developed formal list or dictionary of common software weaknesses. Leveraging the diverse thinking on this topic from academia, the commercial sector, and government, CWE unites the most valuable breadth and depth of content and structure to serve as a standard definition. Our objective is to help shape and mature the code security assessment industry and also dramatically accelerate the use and utility of software assurance capabilities for organizations in reviewing the software systems they acquire or develop.



Working from these collections—as well as those contained in the other information sources listed on the CWE Web site—we developed the current draft of the CWE List, which includes over 600 separate weaknesses and we have created a branding and compliance program to acknowledge and validate tools and services using CWE Identifiers.

Community

The following organizations are actively contributing to the development of CWE: Apple, Cenzec, Core Security,

CWE
cwe.mitre.org

The MITRE Corporation maintains CWE and its public Web site, manages the compatibility program, and provides impartial technical guidance to the CWE Community throughout the process to ensure CWE serves the public interest.

MITRE
202 Burlington Road, Bedford, MA 01730-1420
www.mitre.org

CWE Entries include:

- name of the weakness type
- description of the type
- alternate terms for the weakness
- description of the behavior of the weakness
- description of the exploit of the weakness
- likelihood of exploit for the weakness
- description of the consequences of the exploit
- potential mitigations
- node relationship information
- source taxonomies
- code samples for the languages/architectures
- CVE identifiers of vulnerabilities for which that type of weakness exists
- references

CWE ID 415	Double Free
Description	The product calls free() twice on the same memory address, potentially leading to modification of unexpected memory locations.
Likelihood of Exploit	Low to Medium
Common Consequences	Access control: Doubly freeing memory may result in a write-what-where condition, allowing an attacker to execute arbitrary code.
Potential Mitigations	Architecture and Design: Choose a language that provides automatic memory management. Implementation: Ensure that each allocation is freed only once. After freeing a chunk, set the pointer to NULL to ensure the pointer cannot be freed again. In complicated error conditions, be sure that clean-up routines respect the state of allocation properly. If the language is object oriented, ensure that object destructors delete each chunk of memory only once. Implementation: Use a static analysis tool to find double free instances.
Demonstrative Examples	Example 1: The following code shows a simple example of a double free vulnerability. Double free vulnerabilities have two common (and sometimes overlapping) causes: - Error conditions and other exceptional circumstances - Confusion over which part of the program is responsible for freeing the memory Although some double free vulnerabilities are not much more complicated than the previous example, most are spread out across hundreds of lines of code or even different files. Programmers seem particularly susceptible to freeing global variables more than once. Example 2: While contrived, this code should be exploitable on Linux distributions which do not ship with heap-chunk check summing turned on.
Observed Examples	CVE-2002-0059 - Double free from malformed compressed data. CVE-2003-0545 - Double free from invalid ASN.1 encoding. CVE-2003-1048 - Double free from malformed GIF. CVE-2004-0642 - Double free resultant from certain error conditions. CVE-2004-0772 - Double free resultant from certain error conditions. CVE-2005-0891 - Double free from malformed GIF. CVE-2005-1689 - Double free resultant from certain error conditions.
Node Relationships	Child Of - Operation on Resource in Wrong Phase of Lifetime (666) in View (1000) Child Of - Duplicate Operations on Resource (675) in View (1000) Child Of - Resource Management Errors (399) in View (699) Peer Of - Use After Free (416) in View (699 & 1000) Peer Of - Write-what-where Condition (123) in View (700) Child Of - Indicator of Poor Code Quality (398) in View (700) Child Of - Weaknesses that Affect Memory (633) in View (631) Child Of - CERT C Secure Coding Section 08 – Memory Management (MEM) (742) in View (734) Member Of - Weaknesses Examined by SAMATE (630) in View (630) Peer Of - Signal Handler Race Condition (364) in View (1000)
Source Taxonomies	PLOVER - DFREE - Double-Free Vulnerability 7 Pernicious Kingdoms - Double Free CLASP - Doubly freeing memory CERT C Secure Coding - MEM00-C - Allocate and free memory in the same module, at the same level of abstraction CERT C Secure Coding - MEM01-C - Store a new value in pointers immediately after free() CERT C Secure Coding - MEM31-C - Free dynamically allocated memory exactly once CERT C Secure Coding - MEM00-C - Allocate and free memory in the same module, at the same level of abstraction CERT C Secure Coding - MEM01-C - Store a new value in pointers immediately after free() CERT C Secure Coding - MEM31-C - Free dynamically allocated memory exactly once
Applicable Platforms	C C++
White Box Definitions	A weakness where code path has: 1. start statement that relinquishes a dynamically allocated memory resource 2. end statement that relinquishes the dynamically allocated memory resource

HP, GrammaTech, Klocwork, IBM, Parasoft, Veracode, Symantec, CAST, EC-Council, EMC, Japan's Information-technology Promotion Agency, ISC2, NIST, and Red Hat.

We are also leveraging the work, ideas, and contributions of researchers at Armorize Technologies, Carnegie Mellon's CERT/CC, CERIAS/Purdue University, Cigital, KDM Analytics, Kestrel Technology, Oracle, OWASP, SANS Institute, SkillBridge, UNISYS, WASC, and WhiteHat Security. See the CWE Web site for a complete list of participants and how your organization can contribute.

CWE List

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

The CWE List is currently offered in many views including:

Dictionary - an alphabetic view of the list's enumerated weaknesses

Classification Tree - provides access to individual weaknesses with more simplicity to various potential users through classification layering

Graphical - allows users to better understand individual weaknesses in the classification tree through their broader context and relationships

Slices-by-Topic - provide selective subsets of CWE by language or some other attribute

XML/XSD of CWE content — in toto or by slice — is also available. Additional formats and views will be added in the future. Visit the CWE Web site for the latest information.

MITRE

Learn More – https://cwe.mitre.org