Risk Analysis and Measurement with CWRAF

- Common Weakness Risk Analysis Framework -

April 4, 2012
Making Security Measurable (MSM)

“You Are Here”

Software Assurance  Enterprise Security Management  Threat Management

Design  Deploy  Build  Test  Assess  Design  Test  Deploy

CWE, CAPEC, CWSS, CWRAF  CPE, CCE, OVAL, OCIL, XCCDF, AssetId, ARF  CVE, CWE, CAPEC, MAEC, CybOX, IODEF, RID, RID-T, CYBEX
Today Everything’s Connected – Like an Ecosystem

Your System is attackable...

When this Other System gets subverted through an un-patched vulnerability, a mis-configuration, or an application weakness...
How wetlands work:

- Dissipates stream energy
- Groundwater flow
- Saturated peat stores water
- Bacteria break down contaminants
- Slow release of stored water
- Provides critical wildlife habitat
- Contaminants and sediment are filtered
- Cleaner water outflow

Stream
Microsoft Security Bulletin MS10-071 - Critical
Cumulative Security Update for Internet Explorer (2360131)

Published: October 12, 2010 | Updated: October 13, 2010

Version: 1.1

General Information

Executive Summary
This security update resolves seven privately reported vulnerabilities and three publicly disclosed vulnerabilities in Internet Explorer. The most severe vulnerabilities could allow remote code execution if a user views a specially crafted Web page using Internet Explorer. Users whose accounts are configured to have fewer user rights on the system could be less impacted than users who operate with administrative user rights.

Frequently Asked Questions (FAQ) Related to This Security Update

Vulnerability Information

- Severity Ratings and Vulnerability Identifiers
- AutoComplete Information Disclosure Vulnerability - CVE-2010-0808
- HTML Sanitization Vulnerability - CVE-2010-3243
- HTML Sanitization Vulnerability - CVE-2010-3324
- CSS Special Character Information Disclosure Vulnerability - CVE-2010-3325
- Uninitialized Memory Corruption Vulnerability - CVE-2010-3326
- Anchor Element Information Disclosure Vulnerability - CVE-2010-3327
- Uninitialized Memory Corruption Vulnerability - CVE-2010-3328
- Uninitialized Memory Corruption Vulnerability - CVE-2010-3329
- Cross-Domain Information Disclosure Vulnerability - CVE-2010-3330
- Uninitialized Memory Corruption Vulnerability - CVE-2010-3331
## Oracle Critical Patch Update Advisory - October 2010

**Description**

A Critical Patch Update is a collection of patches for multiple security vulnerabilities. It also includes non-security fixes that are required (because of interdependencies) by those security patches. Critical Patch Updates are cumulative, except as noted below, but each advisory describes only the security fixes added since the previous Critical Patch Update. Thus, prior Critical Patch Update Advisories should be reviewed for information regarding earlier accumulated security fixes. Please refer to:

### Oracle Database Server Risk Matrix

<table>
<thead>
<tr>
<th>CVE-ID</th>
<th>Component</th>
<th>Protocol</th>
<th>Package/Privilege Required</th>
<th>Remote Exploit without Auth.?</th>
<th>CVSS VERSION 2.0 RISK (see Risk Matrix Definitions)</th>
<th>Last Affected Patch set (per Supported Release)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVE-2010-2389</td>
<td>IDM Console</td>
<td>HTTP</td>
<td>None</td>
<td>Yes</td>
<td>7.5 Network Low None Partial+ Partial+ Partial+</td>
<td>10.1.0.5, 10.2.0.3</td>
<td>See Note 1</td>
</tr>
<tr>
<td>CVE-2010-2419</td>
<td>Java Virtual Machine</td>
<td>Oracle Net</td>
<td>Create Session</td>
<td>No</td>
<td>6.5 Network Low Single Partial+ Partial+ Partial+</td>
<td>10.1.0.5, 10.2.0.4, 11.1.0.7, 11.2.0.1</td>
<td>See Note 2</td>
</tr>
<tr>
<td>CVE-2010-1321</td>
<td>Change Data Capture</td>
<td>Oracle Net</td>
<td>Execute on DBMS_CDC_PUBLISH</td>
<td>No</td>
<td>5.5 Network Low Single Partial+ None</td>
<td>-</td>
<td>See Note 2</td>
</tr>
<tr>
<td>CVE-2010-2412</td>
<td>OLAP</td>
<td>Oracle Net</td>
<td>Create Session</td>
<td>No</td>
<td>5.5 Network Low Single Partial+ None</td>
<td>11.1.0.7</td>
<td></td>
</tr>
<tr>
<td>CVE-2010-2415</td>
<td>Change Data Capture</td>
<td>Oracle Net</td>
<td>Execute on DBMS_CDC_PUBLISH</td>
<td>No</td>
<td>4.9 Network Medium Single Partial+ None</td>
<td>-</td>
<td>See Note 2</td>
</tr>
<tr>
<td>CVE-2010-2411</td>
<td>Job Queue</td>
<td>Oracle Net</td>
<td>Execute on SYSDBMS_UJOB</td>
<td>Yes</td>
<td>4.6 Network High Single Partial+ None</td>
<td>-</td>
<td>See Note 2</td>
</tr>
<tr>
<td>CVE-2010-2407</td>
<td>LDK</td>
<td>HTTP</td>
<td>None</td>
<td>Yes</td>
<td>4.3 Network Medium None None None Partial</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>CVE-2010-2391</td>
<td>Core RDBMS</td>
<td>Oracle Net</td>
<td>Create Session</td>
<td>No</td>
<td>3.6 Network High Single Partial+ Partial+ None</td>
<td>-</td>
<td>See Note 2</td>
</tr>
<tr>
<td>CVE-2010-2389</td>
<td>(Oracle Fusion Middleware)</td>
<td>Perl</td>
<td>Local Logon</td>
<td>No</td>
<td>1.0 Local High None Partial+ None</td>
<td>-</td>
<td>See Note 2</td>
</tr>
</tbody>
</table>
## Important: kernel security and bug fix update

<table>
<thead>
<tr>
<th>Advisory</th>
<th>RHSA-2010:0723-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Security Advisory</td>
</tr>
<tr>
<td>Severity</td>
<td>Important</td>
</tr>
<tr>
<td>Issued on</td>
<td>2010-09-29</td>
</tr>
<tr>
<td>Last updated on</td>
<td>2010-09-29</td>
</tr>
<tr>
<td>Affected Products</td>
<td>Red Hat Enterprise Linux (v. 5 server) Red Hat Enterprise Linux Desktop (v. 5 client)</td>
</tr>
<tr>
<td>QVAL</td>
<td>com.redhat.rhsa-20100723.xml</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CVEs (cve.mitre.org)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVE-2010-1083</td>
</tr>
<tr>
<td>CVE-2010-2492</td>
</tr>
<tr>
<td>CVE-2010-2798</td>
</tr>
<tr>
<td>CVE-2010-2938</td>
</tr>
<tr>
<td>CVE-2010-2942</td>
</tr>
<tr>
<td>CVE-2010-2943</td>
</tr>
<tr>
<td>CVE-2010-3015</td>
</tr>
</tbody>
</table>
APPLE-SA-2010-08-11-1 iOS 4.0.2 Update for iPhone and iPod touch

Subject: APPLE-SA-2010-08-11-1 iOS 4.0.2 Update for iPhone and iPod touch
From: Apple Product Security <email@hidden>
Date: Wed, 11 Aug 2010 12:19:43 -0700
Delivered-to: email@hidden
Delivered-to: email@hidden

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

APPLE-SA-2010-08-11-1 iOS 4.0.2 Update for iPhone and iPod touch

iOS 4.0.2 Update for iPhone and iPod touch is now available and addresses the following:

FreeType
CVE-ID: CVE-2010-1797
Available for: iOS 2.0 through 4.0.1 for iPhone 3G and later, iOS 2.1 through 4.2 for iPod touch (2nd generation) and later
Impact: Viewing a PDF document with maliciously crafted embedded fonts may allow arbitrary code execution
Description: A stack buffer overflow exists in FreeType's handling of CFF opcode. Viewing a PDF document with maliciously crafted

-----END PGP SIGNED MESSAGE-----
Vulnerability Type Trends: A Look at the CVE List (2001 - 2007)
Removing and Preventing the Vulnerabilities
Requires More Specific Definitions...CWEs

Improper Neutralization of Input During Web Page Generation (‘Cross-site Scripting’) (79)
- Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS) (80)
- Improper Neutralization of Script in an Error Message Web Page (81)
- Improper Neutralization of Script in Attributes of IMG Tags in a Web Page (82)
- Improper Neutralization of Script in Attributes in a Web Page (83)
- Improper Neutralization of Encoded URI Schemes in a Web Page (84)
- Doubled Character XSS Manipulations (85)
- Improper Neutralization of Invalid Characters in Identifiers in Web Pages (86)
- Improper Neutralization of Alternate XSS Syntax (87)

Improper Restriction of Operations within the Bounds of a Memory Buffer (119)
- Buffer Copy without Checking Size of Input (‘Classic Buffer Overflow’) (120)
- Write-what-where Condition (123)
- Out-of-bounds Read (125)
- Improper Handling of Length Parameter Inconsistency (130)
- Improper Validation of Array Index (129)
- Return of Pointer Value Outside of Expected Range (466)
- Access of Memory Location Before Start of Buffer (786)
- Access of Memory Location After End of Buffer (788)
- Buffer Access with Incorrect Length Value 805
- Untrusted Pointer Dereference (822)
- Use of Out-of-range Pointer Offset (823)
- Access of Uninitialized Pointer (824)
- Expired Pointer Dereference (825)

Path Traversal (22)
- Relative Path Traversal (23)
  - Path Traversal: '../filedir' (24)
  - Path Traversal: '/../filedir' (25)
  - <--------8 more here --------->
  - Path Traversal: '....//' (34)
  - Path Traversal: '.../...//' (35)
- Absolute Path Traversal (36)
  - Path Traversal: '/absolute/pathname/here' (37)
  - Path Traversal: '\absolute\pathname\here' (38)
  - Path Traversal: 'C:dirname' (39)
  - Path Traversal: '\UNC\share\name\ (Windows UNC Share) (40)
Wouldn’t it be nice if the weaknesses in software were as easy to spot and their impact as easy to understand as a screen door in a submarine...
a time-of-check-time-of-use (TOCTOU) bug that led to code calling into a freed memory block. The Common Weakness Enumeration (CWE) classification for this vulnerability is CWE-367.
CWE-367: Time-of-check Time-of-use (TOCTOU) Race Condition

Time-of-check Time-of-use (TOCTOU) Race Condition

Description

Description Summary

The software checks the state of a resource before using that resource, but the resource's state can change between the check and the use in a way that invalidates the results of the check. This can cause the software to perform invalid actions when the resource is in an unexpected state.

Extended Description

This weakness can be security-relevant when an attacker can influence the state of the resource between check and use. This can happen with shared resources such as files, memory, or even variables in multithreaded programs.

Alternate Terms

TOCTTOU: The TOCTTOU acronym expands to "Time Of Check To Time Of Use". Usage varies between TOCTOU and TOCTTOU.

Time of Introduction

- Implementation

Applicable Platforms

Languages

Common Consequences

Scope | Effect
--- | ---
Access Control | The attacker can gain access to otherwise unauthorized resources.
Access Control Authorization | Race conditions such as this kind may be employed to gain read or write access to resources which are not normally readable or writable by the user in question.
Integrity | The resource in question, or other resources (through the corrupted one), may be changed in undesirable ways by a malicious user.
Accountability | If a file or other resource is written in this method, as opposed to in a valid way, logging of the activity may not occur.
Non-Repudiation | In some cases it may be possible to delete files a malicious user might not otherwise have access to, such as log files.
But you also needed to deal with the people that are out there trying to take advantage of vulnerabilities and weaknesses in your technologies, processes, or practices…
...with defensive and offensive security capabilities.
SQL Injection Attack Execution Flow

1. Web Form with ' in all fields
2. One SQL error message
   
SELECT ITEM,PRICE FROM PRODUCT WHERE ITEM_CATEGORY='{$user_input}' ORDER BY PRICE

5. Web Form with: ' exec master..xp_Cmdshell 'dir' --
6. a listing of all directories

© 2012 MITRE
Simple test case for SQL Injection

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
CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

Description

The software constructs all or part of an SQL command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended SQL command when it is sent to a downstream component.

Extended Description

Without sufficient removal or quoting of SQL syntax in user-controllable inputs, the generated SQL query can cause those inputs to be interpreted as SQL instead of ordinary user data. This can be used to alter query logic to bypass security checks, or to insert additional statements that modify the back-end database, possibly including execution of system commands.

SQL injection has become a common issue with database-driven web sites. The flaw is easily detected, and easily exploited, and as such, any site or software package with even a minimal user base is likely to be subject to an attempted attack of this kind. This flaw depends on the fact that SQL makes no real distinction between the control and data planes.

Time of Introduction

- Architecture and Design
- Implementation
- Operation

Applicable Platforms

Languages

All

Technology Classes

Database-Server
<table>
<thead>
<tr>
<th>Rank</th>
<th>Score</th>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>93.8</td>
<td>CWE-89</td>
<td>Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')</td>
</tr>
<tr>
<td>2</td>
<td>83.3</td>
<td>CWE-78</td>
<td>Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')</td>
</tr>
<tr>
<td>3</td>
<td>79.0</td>
<td>CWE-120</td>
<td>Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')</td>
</tr>
<tr>
<td>4</td>
<td>77.7</td>
<td>CWE-79</td>
<td>Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')</td>
</tr>
<tr>
<td>5</td>
<td>76.9</td>
<td>CWE-306</td>
<td>Missing Authentication for Critical Function</td>
</tr>
<tr>
<td>6</td>
<td>76.8</td>
<td>CWE-862</td>
<td>Missing Authorization</td>
</tr>
<tr>
<td>7</td>
<td>75.0</td>
<td>CWE-798</td>
<td>Use of Hard-coded Credentials</td>
</tr>
<tr>
<td>8</td>
<td>75.0</td>
<td>CWE-311</td>
<td>Missing Encryption of Sensitive Data</td>
</tr>
<tr>
<td>9</td>
<td>74.0</td>
<td>CWE-434</td>
<td>Unrestricted Upload of File with Dangerous Type</td>
</tr>
<tr>
<td>10</td>
<td>73.8</td>
<td>CWE-807</td>
<td>Reliance on Untrusted Inputs in a Security Decision</td>
</tr>
<tr>
<td>11</td>
<td>73.1</td>
<td>CWE-250</td>
<td>Execution with Unnecessary Privileges</td>
</tr>
<tr>
<td>12</td>
<td>70.1</td>
<td>CWE-352</td>
<td>Cross-Site Request Forgery (CSRF)</td>
</tr>
<tr>
<td>13</td>
<td>69.3</td>
<td>CWE-22</td>
<td>Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')</td>
</tr>
<tr>
<td>14</td>
<td>68.5</td>
<td>CWE-494</td>
<td>Download of Code Without Integrity Check</td>
</tr>
<tr>
<td>15</td>
<td>67.8</td>
<td>CWE-863</td>
<td>Incorrect Authorization</td>
</tr>
</tbody>
</table>
Mass SQL Injection Attacks Uses Automated Tools, Search to Infect New Sites

By: Fahmida Y. Rashid
2012-01-10
Article Rating:☆☆☆☆☆ / 0

There are user comments on this IT Security & Network Security News & Reviews story.

Attackers are using search results as a reconnaissance tool to identify sites to hit in the latest mass injection attack directing users to Lilupophilupop.com.

Security researchers monitoring mass SQL injection attacks warned the latest one may be nearing a million infected pages using a combination of automated tools and reconnaissance using search engines.

The "Lilupophilupop" SQL injection campaign has infected a little over a million URLs since it was first detected in early December, according to a post on the SANS Institute's Internet Storm Center. The security firm detected only 80 corrupted URLs when it first noticed the campaign. Mark Hofman, a handler at the SANS Institute's Internet Storm, acknowledged the list contained duplicate URLs but regardless of the actual number of infected sites, the campaign was definitely growing.

Victims who land on the infected URLs are redirected to other sites and wind up on Lilupophilupop.com, which can display an "adobeflash page" where they are encouraged to download what they think is an update to Adobe Flash, or to a fake antivirus site. The scam's ultimate goal is to trick victims into paying for software or antivirus protection they don't need, and will likely cause more problems once installed.

"Sources of the attack vary, it is automated and spreading fairly rapidly," Hofman wrote in an initial analysis of the attack.

This newest mass injection is similar to the LizaMoon attack, which was responsible for redirecting 1.5 million URLs to fake antivirus pages. Websites based in the Netherlands are the biggest victims of Lilupophilupop, followed by French sites, according to the SANS Institute. Sites with backends running on IIS, ASP or Microsoft SQL Server seem to be the primary target.
Test and vulnerability assessment

Testing applications for security defects should be an integral and organic part of any software testing process. During security testing, organizations should test to help ensure that the security requirements have been implemented and the product is free of vulnerabilities.

The SEF refers to the MITRE Common Weakness Enumeration\(^5\) (CWE) list and the Common Vulnerability Baseline\(^6\) to be tested. This helps information and services are protected against the most common threats.

Creating a secure software development plan includes:

- Pre-developing software and systems against known exploits.
- Testing software and systems against known vulnerabilities.
- Defining and implementing software and systems against identified risks.
- Developing software and systems that are robust against identified threats.
- Defining and testing software and systems that are resilient against identified risks.
- Defining and testing software and systems that are resilient against identified threats.

Resources available to help organizations protect systems in development:

<table>
<thead>
<tr>
<th>Resource</th>
<th>Focus</th>
</tr>
</thead>
<tbody>
<tr>
<td>DoD Information Assurance Certification and Accreditation Process (DIACAP)</td>
<td>The DIACAP defines the minimum standards for security controls, but it is up to organizations to use the guidance to develop effective controls.</td>
</tr>
<tr>
<td>Defense Information Systems Agency (DISA)</td>
<td>The DISA provides guidelines for developing and implementing an information assurance plan.</td>
</tr>
<tr>
<td>The Common Weakness Enumeration project, a community-based program sponsored by the MITRE Corporation, an IBM Business Partner</td>
<td>The MITRE Corporation maintains the online common vulnerabilities and exposures (CVE) database.</td>
</tr>
<tr>
<td>OWASPs</td>
<td>One of the best sources for information on web application security issues, the OWASP Top 10 list of the most dangerous and commonly found and commonly exploited vulnerabilities.</td>
</tr>
<tr>
<td>BSIMM</td>
<td>Created by Cigital, an IBM Business Partner, the BSIMM is designed to help organizations and plan a software security initiative.</td>
</tr>
<tr>
<td>IBM X-Force research and development team</td>
<td>A global cybersecurity team that monitors threats and attacks and provides assessments on the latest threats and vulnerabilities.</td>
</tr>
<tr>
<td>IBM Institute for Advanced Security (IAS)</td>
<td>This companywide cybersecurity initiative applies IBM research, services, and technologies to help organizations assess and mitigate cyber risks and vulnerabilities.</td>
</tr>
</tbody>
</table>

Creating a secure software development plan includes:

- Pre-developing software and systems against identified threats.
- Testing software and systems against identified vulnerabilities.
- Defining and implementing software and systems against identified risks.
- Developing software and systems that are robust against identified threats.
- Defining and testing software and systems that are resilient against identified risks.
- Defining and testing software and systems that are resilient against identified threats.
Making the Business Case for Software Assurance

Nancy R. Mead
Julia P. Allen
W. Arthur Conklin
Antonio Drommi
John Harrison
Jeff Ingalske
James Rainey
Dan Shoemake

April 2009

SPECIAL REPORT
CMUSEI-2009-SR-001

CERT Program
Unrestricted distribution subject to the copyright.

http://www.cert.cmu.edu

Carnegie Mellon

OVM: An Ontology for Vulnerability Management
Ju An Wang & Minzhe Guo
Southern Polytechnic State University
1100 South Marietta Parkway
Marietta, GA 30060
(01) 478-915-3718
jwang@spsu.edu

ABSTRACT
In order to reach the goals of the Information Security Automation Program (ISAP) [1], we propose an ontological approach to supporting and utilizing the fundamental concepts in information security and their relationship, retrieving vulnerability data and reasoning about the cause and impact of vulnerabilities. Our ontology for vulnerability management (OVM) has been populated with all vulnerabilities in NVD [2] with additional reference rules, knowledge representations, and data-mining mechanisms. With the seamless integration of common vulnerabilities and their related concepts such as attacks and countermeasures, OVM provides a promising pathway to making ISAP succeed.

Categories and Subject Descriptors
C.2.0 [Computer-Communication Networks]: General [Security and protection]; K.6.5 [Management of Computing and Information Systems]: Security and Protection;

General Terms
Ontology, Security, Vulnerability Analysis and Management

Keywords
Security, vulnerability, Semantic technology, Ontology, Vulnerability analysis

1. INTRODUCTION
The Information Security Automation Program (ISAP) is a U.S. government multi-agency initiative to enable automation and standardization of technical security operations [1]. Its high-level goals include standards-based automation of security checking and verification as well as automation of technical compliance activities. In low-level objectives include enabling standards-based communication of vulnerability data, outsourcing and managing configuration baselines for various IT products, assessing mission assurance and reporting compliance status, using standardized metrics to weight and aggregate potential vulnerabilities, and reasoning about vulnerabilities [1]. Secure computer systems ensure that confidentiality, integrity, and availability are maintained for users, data, and other information. Over the past a few decades, a significantly large amount of knowledge has been accumulated in the area of information security. However, a lot of concepts in information security are vaguely defined and sometimes they have different semantics in different contexts, causing misunderstanding among stakeholders due to the language ambiguity. On the other hand, the naming, design and development of security tools [1-5] require a systematic classification and definition of security concepts and techniques. It is important to have a clearly defined vocabulary and standardized language as means to accurately communicate system vulnerability information and their countermeasures among all the people involved. We believe that semantic technology in general, and ontology in particular, could be a useful tool for system security. Our research work has confirmed this belief and this paper will report some of our work in this area.

An ontology is a specification of concepts and their relationships. Ontology represents knowledge in a formal and structured form. Therefore, ontology provides a better tool for communication, reasoning, and organization of knowledge. Ontology is a knowledge representation (KR) system based on Description Logics (DL) [6], which is an umbrella name for a family of KR formalizations representing knowledge in various domains. The DL formation specifies a knowledge domain as the “world” by first defining the relevant concepts of the domain, and then uses these concepts to specify properties of objects and individuals occurring in the domain [10-12]. Semantic technologies not only provide a tool for communication, but also a foundation for high-level reasoning and decision-making. Ontology, in particular, provides the potential of formal logical inference based on well-defined data and knowledge bases. Ontology captures the relationships between collected data and the explicit knowledge of concepts and relationships to deduce the implicit and inherent knowledge. As a matter of fact, a heavy-weight ontology could be defined as a formal logic system, as it includes facts and rules, concepts, concept taxonomies, relationships, axioms, axioms and constraints.

A vulnerability is a security flaw, which arises from computer design, implementation, maintenance, and operation. Research in the area of vulnerability analysis focuses on discovery of previously unknown vulnerabilities and quantification of the security of systems according to some metrics. Researchers at NASA have provided a standard format for naming a security vulnerability, called Common Vulnerabilities and Exposures (CVE) [14], which assigns each vulnerability a unique identification number. We have designed a vulnerability ontology (OVM) for vulnerability management) populated with all existing vulnerabilities in NVD [2]. It supports research on reasoning about vulnerabilities and characterization of vulnerabilities and their impact on computing systems. Various researchers can use our ontology in support of vulnerability analysis, tool development and vulnerability management.

The rest of this paper is organized as follows: Section 2 presents the architecture of our OVM. Section 3 discusses how to populate the OVM with vulnerability instances from NVD and other
A complete body of knowledge covering the entire field of software engineering may be years away. However, the body of knowledge needed by professionals to create software free of common and critical security flaws has been developed, vetted widely and kept up to date. That is the foundation for a certification program in software assurance that can gain wide adoption. It was created in late 2008 by a consortium of national experts, sponsored by DHS and NSA, and was updated in late 2009. It contains ranked lists of the most common errors, explanations of why the errors are dangerous, examples of those errors in multiple languages, and ways of eliminating those errors. It can be found at http://cwe.mitre.org/top25.

Any programmer who writes code without being aware of those problems and is not capable of writing code free of those errors is a threat to his or her employers and to others who use computers connected to systems running his or her software.
The Certified Secure Software Lifecycle Professional (CSSLP) Certification Program will show software lifecycle stakeholders not only how to implement security, but how to glean security requirements, design, architect, test and deploy secure software.

An Overview of the Steps:

(ISC)² 5-day CSSLP CBK® Education Program
Educate yourself and learn security best practices and industry standards for the software lifecycle through the CSSLP Education Program. (ISC)² provides education your way to fit your life and schedule. Completing this course will, not only teach all of the

COMPUTER BASED TESTING
NOW AVAILABLE FOR THE
CSSLP
The paper also contains two important, additional sections for each listed practice that will further increases its value to implementers—Common Weakness Enumeration (CWE) references and Verification guidance.
Why Does Code Have Vulnerabilities?

MITRE has catalogued almost 700 different kinds of software weaknesses in their CWE project. There are all different ways that software developers can make mistakes that lead to insecurity. Every one of these weaknesses is subtle and many are seriously tricky. Software developers are not taught about these weaknesses in school and most do not receive any training on the job about these problems.

These problems have become so important in recent years because we continue to increase connectivity and to add technologies and protocols at a shocking rate. Our ability to invent technology has seriously outstripped our ability to secure it. Many of the technologies in use today simply have not received any security scrutiny.

There are many reasons why businesses are not spending the appropriate amount of time on security. Ultimately, these reasons stem from an underlying problem in the software market. Because software is essentially a black-box, it is extremely difficult to tell the difference between good code and insecure code. Without this visibility, buyers won’t pay more for secure code, and vendors would be foolish to spend extra effort to produce secure code.

This lack of visibility is the reason why we advocate for security code review. Here are some of the (unjustified) excuses that we hear for not putting more effort into security:

- “We never get hacked (that I know of), so we don’t need security.”
- “Our customers would never ask for more security.”
- “It’s too expensive to improve security.”
- “We have enough time to do it.”
- “Our customers don’t care about security.”

Nevertheless, we still frequently get pushback when we advocate for security code review. Here are some of the (unjustified) excuses that we hear for not putting more effort into security:
## Threat Classification Taxonomy Cross Reference View

This view contains a mapping of the WASC Threat Classification's Attacks and Weaknesses with MITRE's Common Weakness Enumeration, MITRE's Common Attack Pattern Enumeration and Classification, OWASP Top Ten 2010 R1 (original mapping with OWASP Top Ten from Jeremiah Grossman & Bill Conlon) and SANS/CWE and OWASP Top Ten 2007 and 2004 (original mapping from Dan Cornell, Denim Group).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WASC-02</td>
<td>Insufficient Authentication</td>
<td>268</td>
<td></td>
<td>253</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WASC-03</td>
<td>Integer OVERflows</td>
<td>190</td>
<td>118</td>
<td>682</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WASC-04</td>
<td>Insufficient Transport Layer Protocols</td>
<td>311</td>
<td>523</td>
<td>119</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WASC-05</td>
<td>Remote File Inclusion</td>
<td>98</td>
<td>131</td>
<td>253 626</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WASC-06</td>
<td>Format String</td>
<td>114</td>
<td></td>
<td>67</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WASC-07</td>
<td>Buffer Overflows</td>
<td>119 120</td>
<td>10 160</td>
<td>119</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WASC-08</td>
<td>Cross-site Scripting</td>
<td>29</td>
<td>18 19 63</td>
<td>79</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WASC-09</td>
<td>Cross-site Request Forgery</td>
<td>312</td>
<td></td>
<td>62 152</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WASC-10</td>
<td>Cross-site Request Forgery</td>
<td>312</td>
<td></td>
<td>62 152</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[View on MITRE](http://projects.webappsec.org/w/page/13246075/Threat-Classification-Taxonomy-Cross-Reference-View)
- The way how the **CAPEC and related CWE taxonomies** are to be used by the developer, which needs to consider and provide sufficient and effective mitigation to all applicable attacks and weaknesses.

- The way how the CAPEC and related CWE taxonomies are to be used by the evaluator, which needs to consider all the applicable attack patterns and be able to exploit all the related software weaknesses while performing the subsequent AVA_VAN activities.

- How incomplete entries from the CAPEC are to be addressed during an evaluation.

- How to incorporate to the evaluation attacks and weaknesses not included in the CAPEC.
CWE Compatibility & Effectiveness Program
(launched Feb 2007)

First 13 CWE Compatible Certificates Awarded
28 Feb 2012
[비즈니스 임팩트를 줄여주는 새로운 품질 관리 방법론]

y5를 사용하여, 소프트웨어 결함을 없애는 5가지 스텝은 아래와 같습니다.

1. 스캔 소프트웨어
2. 검출 결함 우선순위
3. [그림]
4. 수정 우선순위는 결합부에
5. 리포트 결합결정사항

「ビジネスインパクトから考える新しい品質管理」

Coverity5を使用して、ソフトウェア不具合を簡単に除去する 5ステップは以下の通りです。

1. スキャン ソフトウェア
2. 検出 不具合に対する優先度
3. マッピング ビジネスへの影響
4. 修正 優先順位の高い不具合
5. レポート 不具合修正状況
CWE Coverage – Implemented...

CWE Coverage for Common Weakness Enumeration (CWE): Java

CWE Coverage For Common Weakness Enumeration (CWE): C/C++

CENZIC Product Suite is CWE Compatible

CWE IDs mapped to Klocwork Java issue types

CWE IDs mapped to Klocwork C and C++ issue types/ja
Recreation Use
Power Use
Agricultural Use
Industrial Use
Home Use
Recreation Use
Agricultural Use
Groundwater
Riparian Habitat
Prioritizing weaknesses to be mitigated

Lists are a good start but they are designed to be broadly applicable

We would like a way to specify priorities based on business/mission risk
Common Weakness Risk Analysis Framework (CWRAF)

*How do I identify which of the 800+ CWE’s are most important for my specific business domain, technologies and environment?*

Common Weakness Scoring System (CWSS)

*How do I rank the CWE’s I care about according to my specific business domain, technologies and environment?*

How do I identify and score weaknesses important to my organization?
CWRAF-Level Technical Impacts

1. Modify data
2. Read data
3. DoS: unreliable execution
4. DoS: resource consumption
5. Execute unauthorized code or commands
6. Gain privileges / assume identity
7. Bypass protection mechanism
8. Hide activities
Common Weakness Risk Analysis Framework (CWRAF)

Technical Impacts

1. Modify data
2. Read data
3. DoS: unreliable execution
4. DoS: resource consumption
5. Execute unauthorized code or commands
6. Gain privileges / assume identity
7. Bypass protection mechanism
8. Hide activities

Weightings

W1=0  
W2=0  
W3=1  
0  
W4=4  
W5=1  
0  
W6=0  
W7=0  
W8=0

Layers

1. System
2. Application
3. Network
4. Enterprise

Technical Impact Scorecard

Multiple pieces – we’ll focus on “Vignettes”
CWRAF: Technical Impact Scorecard

For each layer and each technical impact assign a weighting from 0 to 10.

<table>
<thead>
<tr>
<th>Layer</th>
<th>MD</th>
<th>RD</th>
<th>UE</th>
<th>RC</th>
<th>EA</th>
<th>GP</th>
<th>BP</th>
<th>HA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System</td>
<td></td>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Enterprise</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CWRAF: Technical Impact Scorecard

<table>
<thead>
<tr>
<th></th>
<th>MD</th>
<th>RD</th>
<th>UE</th>
<th>RC</th>
<th>EA</th>
<th>GP</th>
<th>BP</th>
<th>HA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>9</td>
<td>7</td>
<td>3</td>
<td>2</td>
<td>10</td>
<td>8</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>System</td>
<td>8</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>10</td>
<td>9</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>Network</td>
<td>9</td>
<td>5</td>
<td>6</td>
<td>2</td>
<td>10</td>
<td>5</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>Enterprise</td>
<td>4</td>
<td>7</td>
<td>6</td>
<td>2</td>
<td>10</td>
<td>6</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

These weightings can now be used to evaluate individual CWE’s based on each CWE’s Technical Impacts

Note: Values for illustrative purposes only
<table>
<thead>
<tr>
<th></th>
<th>MD</th>
<th>RD</th>
<th>UE</th>
<th>RC</th>
<th>EA</th>
<th>GP</th>
<th>BP</th>
<th>HA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>9</td>
<td>7</td>
<td>3</td>
<td>2</td>
<td>10</td>
<td>8</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>System</td>
<td>8</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>10</td>
<td>9</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>Network</td>
<td>9</td>
<td>5</td>
<td>6</td>
<td>2</td>
<td>10</td>
<td>5</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>Enterprise</td>
<td>4</td>
<td>7</td>
<td>6</td>
<td>2</td>
<td>10</td>
<td>6</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

CWE-78 Technical Impacts

**CWSS Score for CWE-78 for this vignette**

95
<table>
<thead>
<tr>
<th>CWSS Score</th>
<th>CWE</th>
</tr>
</thead>
<tbody>
<tr>
<td>97</td>
<td>CWE-79</td>
</tr>
<tr>
<td>95</td>
<td>CWE-78</td>
</tr>
<tr>
<td>94</td>
<td>CWE-22</td>
</tr>
<tr>
<td>94</td>
<td>CWE-434</td>
</tr>
<tr>
<td>94</td>
<td>CWE-798</td>
</tr>
<tr>
<td>93</td>
<td>CWE-120</td>
</tr>
<tr>
<td>93</td>
<td>CWE-250</td>
</tr>
<tr>
<td>92</td>
<td>CWE-770</td>
</tr>
<tr>
<td>91</td>
<td>CWE-829</td>
</tr>
<tr>
<td>91</td>
<td>CWE-190</td>
</tr>
<tr>
<td>91</td>
<td>CWE-494</td>
</tr>
<tr>
<td>90</td>
<td>CWE-134</td>
</tr>
<tr>
<td>90</td>
<td>CWE-772</td>
</tr>
<tr>
<td>90</td>
<td>CWE-476</td>
</tr>
<tr>
<td>90</td>
<td>CWE-131</td>
</tr>
</tbody>
</table>

**CWSS Scoring Engine**

- User-defined cutoff

**Most Important Weaknesses**

- "Vignette"
Scoring Weaknesses Discovered in Code using CWSS

Steps:
1. Establish weightings for the vignette
2. Run code through analysis tool(s)
3. Tools produce report of CWE’s found in code
4. CWSS scoring engine automatically scores each CWE based on vignette definition
5. Go to step 2 for each piece of code applicable to this vignette

Line 23: CWE-109
Line 72: CWE-84
Line 104: CWE-482
Line 212: CWE-9
Line 213: CWE-754
...

Line 212: CWE-9: 9.9
Line 72: CWE-84: 7.9
Line 23: CWE-109: 5.6
Line 104: CWE-482: 3.1
Line 213: CWE-754: 0.0
...

Step 1 is only done once – the rest is automatic
Organizations that have declared plans to support CWSS in their future offerings and are working with MITRE to help evolve CWSS to meet their customer's and the community's needs for a scoring system for software errors.
CWE Coverage Claims Representation

Set of CWE’s tool *claims* to cover

Tool A

Tool B

Tool C

Most Important Weaknesses (CWE’s)

Which static analysis tools find the CWE’s I care about?
CWSS for a Technology Group

Web Application Technology Group

CWE Top 10 List for Web Applications can be used to:

- Identify skill and training needs for your web team
- Include in T’s & C’s for contracting for web development
- Identify tool capability needs to support web assessment
<table>
<thead>
<tr>
<th>Domain Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E-Commerce</td>
<td>The use of the Internet or other computer networks for the sale of products and services, typically using on-line capabilities.</td>
</tr>
<tr>
<td>Banking &amp; Finance</td>
<td>Financial services, including banks, stock exchanges, brokers, investment companies, financial advisors, and government regulatory agencies.</td>
</tr>
<tr>
<td>Public Health</td>
<td>Health care, medical encoding and billing, patient information/data, critical or emergency care, medical devices (implantable, partially embedded, patient care), drug development and distribution, food processing, clean water treatment and distribution (including dams and processing facilities), etc.</td>
</tr>
<tr>
<td>Energy</td>
<td>Smart Grid (electrical network through a large region, using digital technology for monitoring or control), nuclear power stations, oil and gas transmission, etc.</td>
</tr>
<tr>
<td>Chemical</td>
<td>Chemical processing and distribution, etc.</td>
</tr>
<tr>
<td>Manufacturing</td>
<td>Plants and distribution channels, supply chain, etc.</td>
</tr>
<tr>
<td>Shipping &amp; Transportation</td>
<td>Aerospace systems (such as safety-critical ground aviation systems, on-board avionics, etc), shipping systems, rail systems, etc.</td>
</tr>
<tr>
<td>National Security</td>
<td>National security systems (including networks and weapon systems), Defense Industrial Base, etc.</td>
</tr>
<tr>
<td>Government and Commercial Security</td>
<td>Homeland Security systems, commercial security systems, etc.</td>
</tr>
<tr>
<td>Emergency Services</td>
<td>Systems and services that support first responders, incident management and response, law enforcement, and emergency services for citizens, etc.</td>
</tr>
<tr>
<td>Telecommunications</td>
<td>Cellular services, land lines, VOIP, cable &amp; fiber networks, etc.</td>
</tr>
<tr>
<td>Telecommuting &amp; Teleworking</td>
<td>Support for employees to have remote access to internal business networks and capabilities.</td>
</tr>
<tr>
<td>eVoting</td>
<td>Electronic voting systems, as used within state-run elections, shareholder meetings, etc.</td>
</tr>
<tr>
<td>Technology Group</td>
<td>Archetypes/Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Web Applications</td>
<td>Web browser, web-server, web-based applications and services, etc.</td>
</tr>
<tr>
<td>Industrial Control Systems</td>
<td>SCADA, process control system, etc.</td>
</tr>
<tr>
<td>Real-time, Embedded Systems</td>
<td>Embedded Device, Programmable logic controller, implanted medical devices, avionics package.</td>
</tr>
<tr>
<td>End-point Computing Devices</td>
<td>Smart phone, laptop, personal digital assistant (PDA), and other remote devices that leave the enterprise and/or connect remotely to the enterprise.</td>
</tr>
<tr>
<td>Cloud Computing</td>
<td>Hosted applications or capabilities provided over the Internet, including Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure as a Service (IaaS).</td>
</tr>
<tr>
<td>Operating Systems</td>
<td>General-purpose OS, virtualized OS, Real-time operating system (RTOS), hypervisor, microkernel.</td>
</tr>
<tr>
<td>Enterprise Desktop Applications/Systems</td>
<td>Office products such as word processing, spreadsheets, project management, etc.</td>
</tr>
</tbody>
</table>
Common Weakness Risk Assessment Framework uses Vignettes with Archetypes to identify top CWEs in respective Domain/Technology Groups
Organizations that have declared plans to work on CWRAF Vignettes and Technical Scorecards with MITRE to help evolve CWRAF to meet their customer's and the community's needs for a scoring system for software errors.
Relationships between CWRAF, CWSS, and CWE

**CWRAF**
- Provides Vignettes (technical & business context) to specify relevant, applicable CWE IDs

**CWEs (by ID)**
- CWE 79
- CWE 120
- CWE 78
- CWE 352
- CWE 434
- CWE 22
- CWE 89
- CWE 311
- CWE 285
- CWE 807

**CWSS**
- Applies Scoring Criteria to Rank Relevant Weaknesses
- Influences Scoring Using Business Value and Technical Context
- Influences Scoring Using Technical Impacts

**Note:** CWSS can be used in the context of CWRAF; but it is not a requirement.
Contact Info

cwss@mitre.org

cwe@mitre.org