HS SEDI
Homeland Security Systems Engineering and Development Institute

Knowing Your Weaknesses: The (CWE) Initiative

Bob Martin

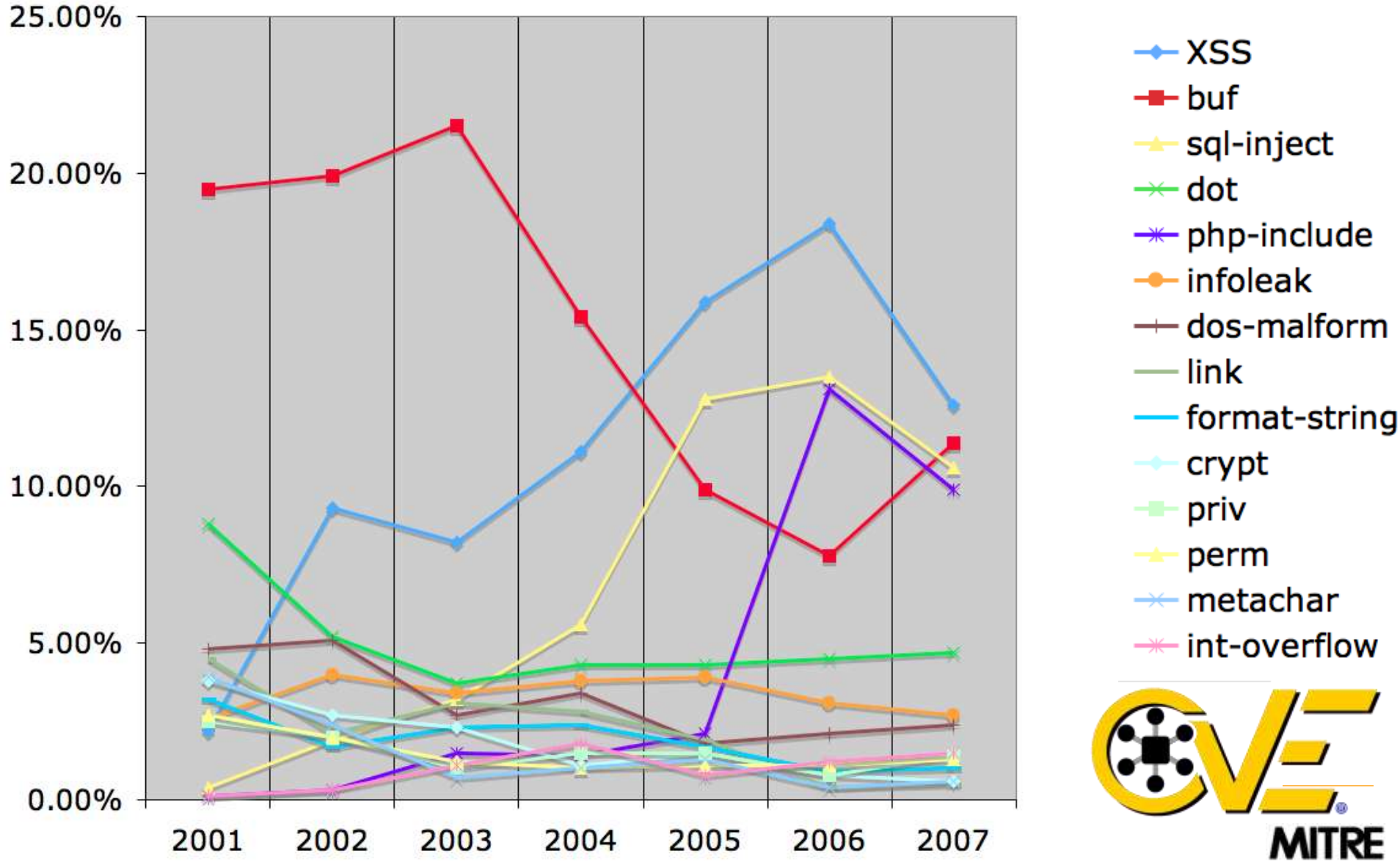
September 27, 2010

Homeland Security


If the weaknesses in software were as easy to spot and their impact as obvious as…
Missing Authentication for Critical Function (CWE-306)
Using Unpublished Web Service APIs (CAPEC-36)

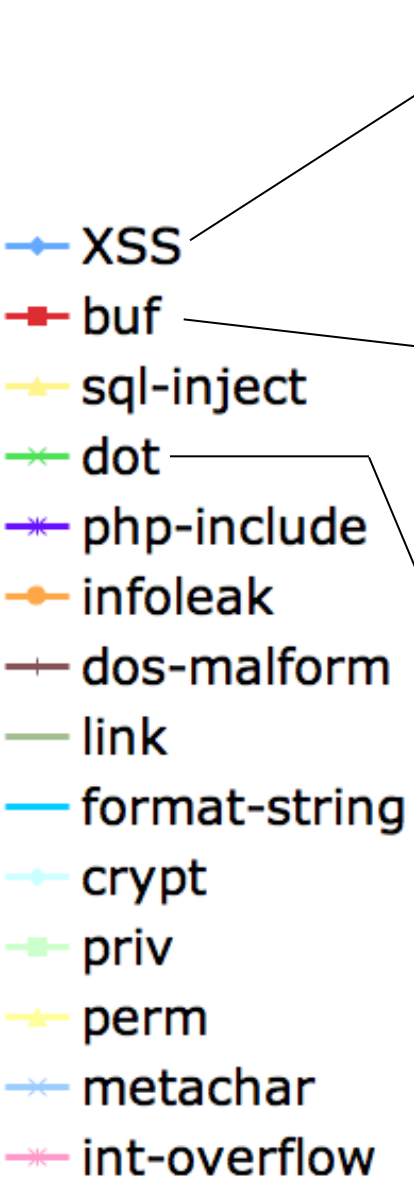
Vulnerability Type Trends:
A Look at the CVE List (2001 - 2007)

HS SEDI
Homeland Security Systems Engineering and Development Institute



CVE
MITRE

Removing and Preventing the Vulnerabilities Requires More Specific Definitions…CWEs



Failure to Sanitize Directives in a Web Page (aka 'Cross-site scripting' (XSS)) (79)
- Failure to Sanitize Script-Related HTML Tags in a Web Page (Basic XSS) (80)
- Failure to Sanitize Directives in an Error Message Web Page (81)
- Failure to Sanitize Script in Attributes of IMG Tags in a Web Page (82)
- Failure to Sanitize Script in Attributes in a Web Page (83)
- Failure to Resolve Encoded URI Schemes in a Web Page (84)
- Doubled Character XSS Manipulations (85)
- Invalid Characters in Identifiers (86)
- Alternate XSS syntax (87)

Failure to Constrain Operations within the Bounds of an Allocated Memory Buffer (119)
- Unbounded Transfer (‘Classic Buffer Overflow’) (120)
- Write-what-where Condition (123)
- Boundary Beginning Violation (’Buffer Underwrite') (124)
- Out-of-bounds Read (125)
- Wrap-around Error (128)
- Unchecked Array Indexing (129)
- Incorrect Calculation of Buffer Size (131)
- Miscalculated Null Termination (132)
- Return of Pointer Value Outside of Expected Range (466)

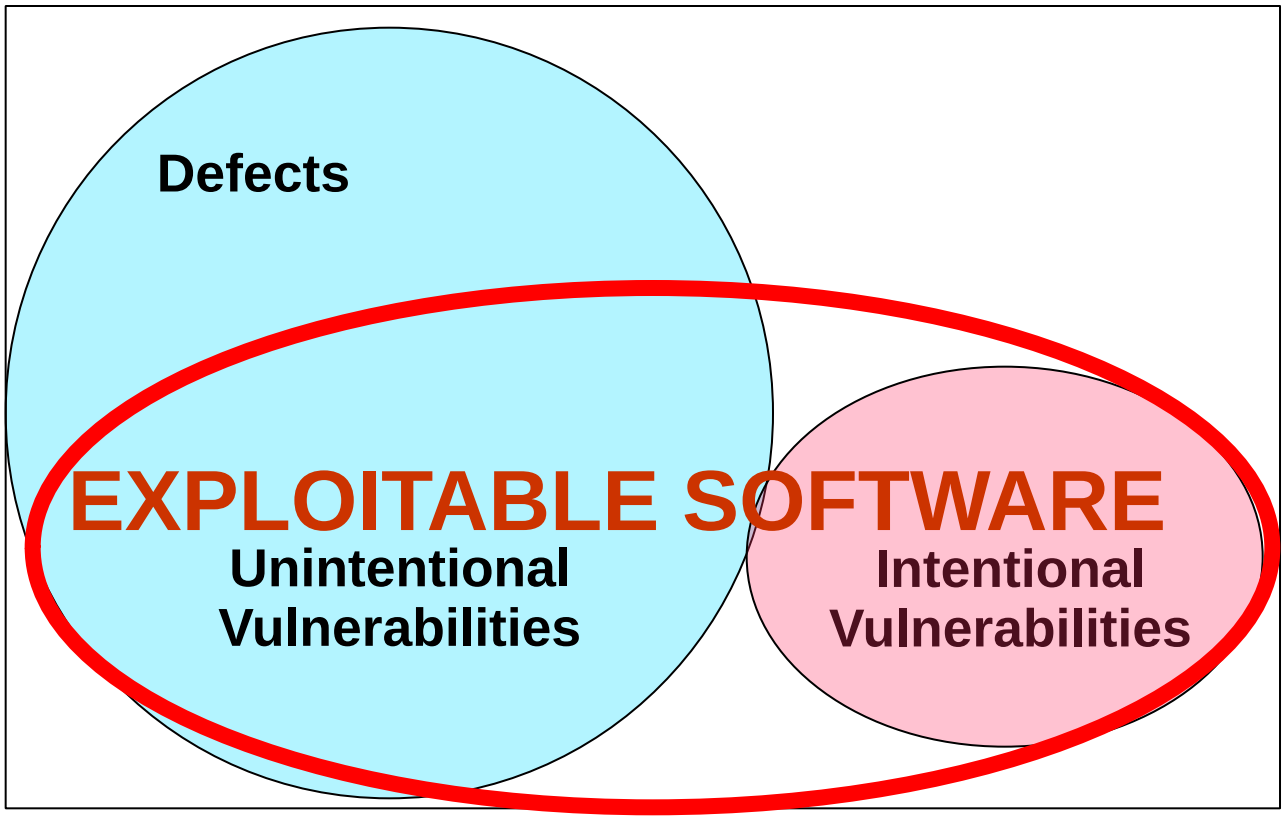
Path Traversal (22)
- Relative Path Traversal (23)
  - Path Traversal: '\..\filename' (29)
  - Path Traversal: '\dir\..\filename' (30)
  - Path Traversal: 'dir\..\filename' (31)
  - Path Traversal: '...' (Triple Dot) (32)
  - Path Traversal: '....' (Multiple Dot) (33)
  - Path Traversal: '....//' (34)
  - 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)

Exploitable Software Weaknesses (a.k.a. Vulnerabilities)

Vulnerabilities can be the outcome of non-secure practices and/or malicious intent of someone in the development/support lifecycle.

The exploitation potential of a vulnerability is independent of the "intent" behind how it was introduced.

DEPLOYED SOFTWARE



Intentional vulnerabilities are spyware & malicious logic deliberately imbedded (and might not be considered defects but they can make use of the same weakness patterns as unintentional mistakes)

Note: Chart is not to scale – notional representation -- for discussions

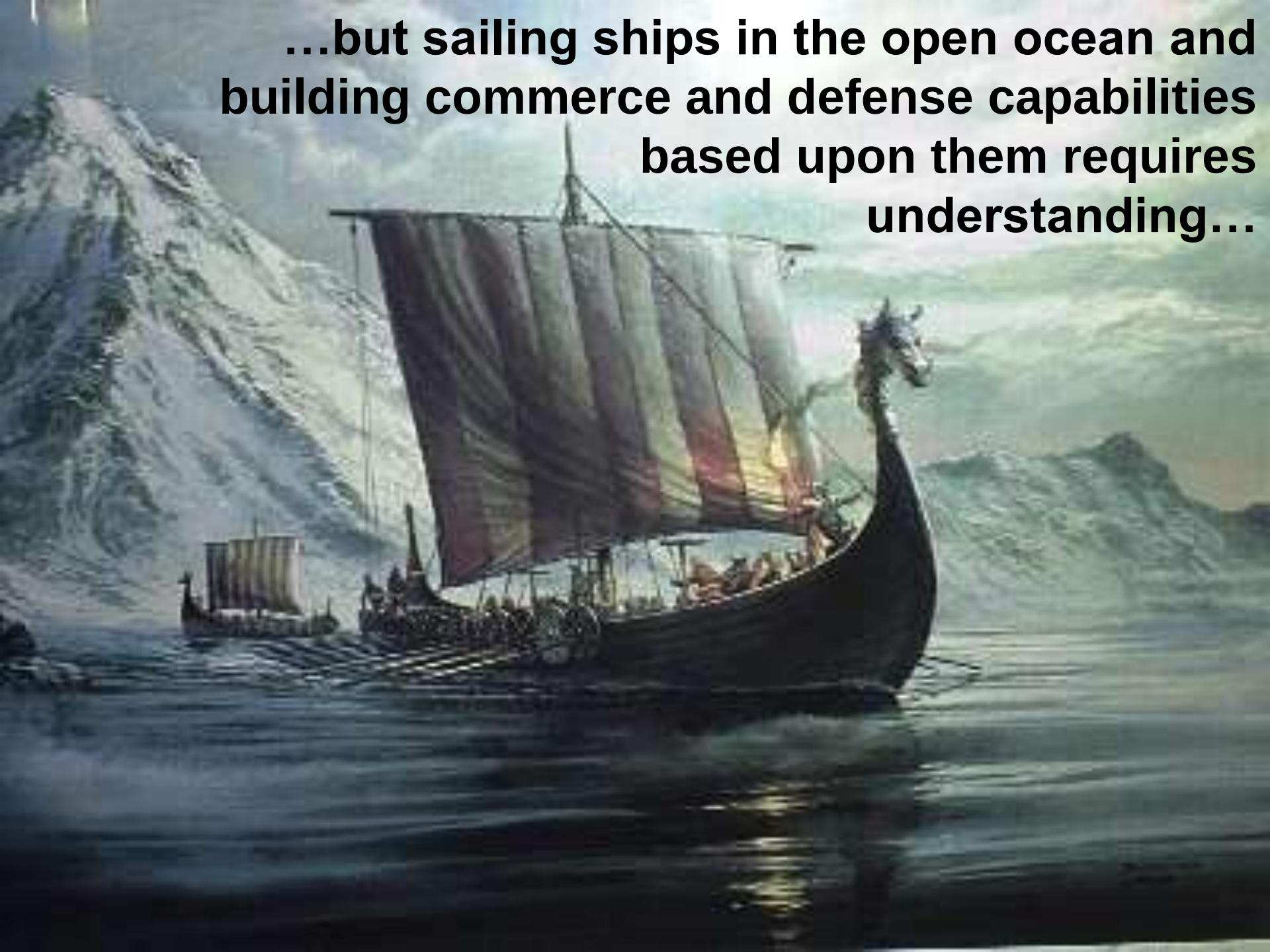
Common Weakness Enumeration (CWE)

HS SEDI
Homeland Security Systems Engineering and Development Institute

- dictionary of weaknesses
  - weaknesses that can lead to exploitable vulnerabilities (i.e. CVEs)
  - the things we don't want in our code, design, or architecture
  - web site with XML of content, sources of content, and process used
- structured views
  - currently provide hierarchical view into CWE dictionary content
  - will evolve to support alternate views
- open community process
  - to facilitate common terms/ concepts/facts and understanding
  - allows for vendors, developers, system owners and acquirers to understand tool capabilities/ coverage and priorities
  - utilize community expertise

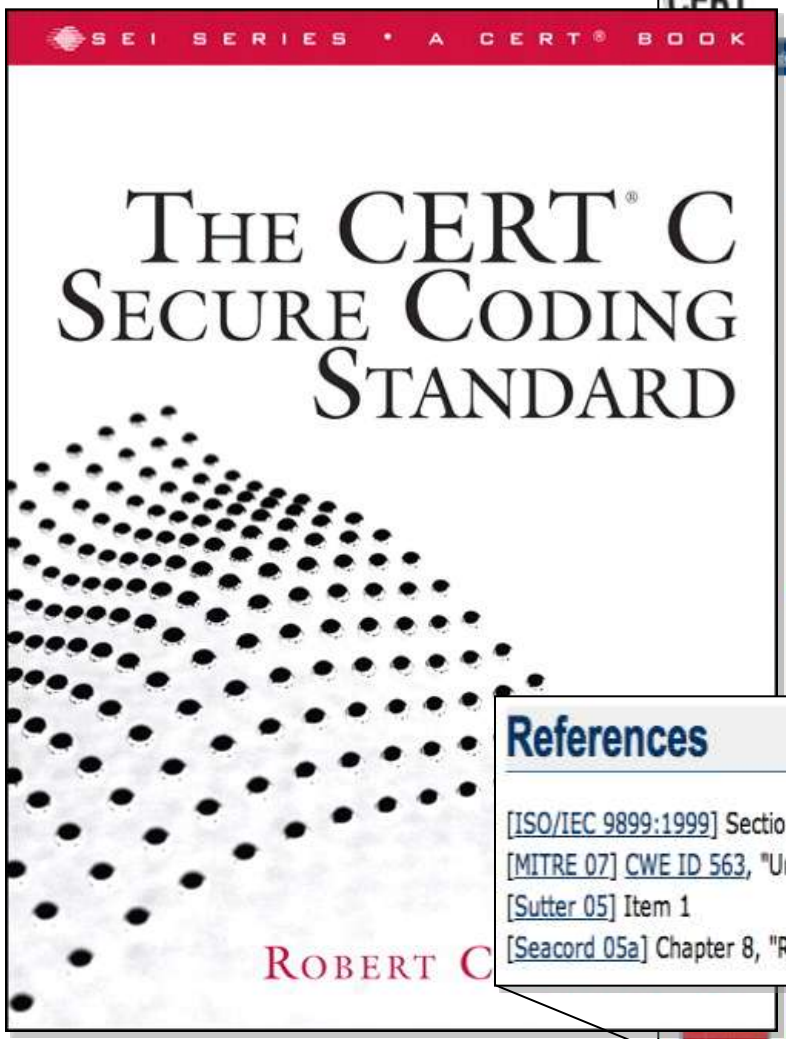
Foundation for other DHS, NSA, OSD, NIST, OWASP, SANS, and OMG SwA Efforts


Building Software
only require a few
skills and basic
understanding…



…but sailing ships in the open ocean and building commerce and defense capabilities based upon them requires understanding…

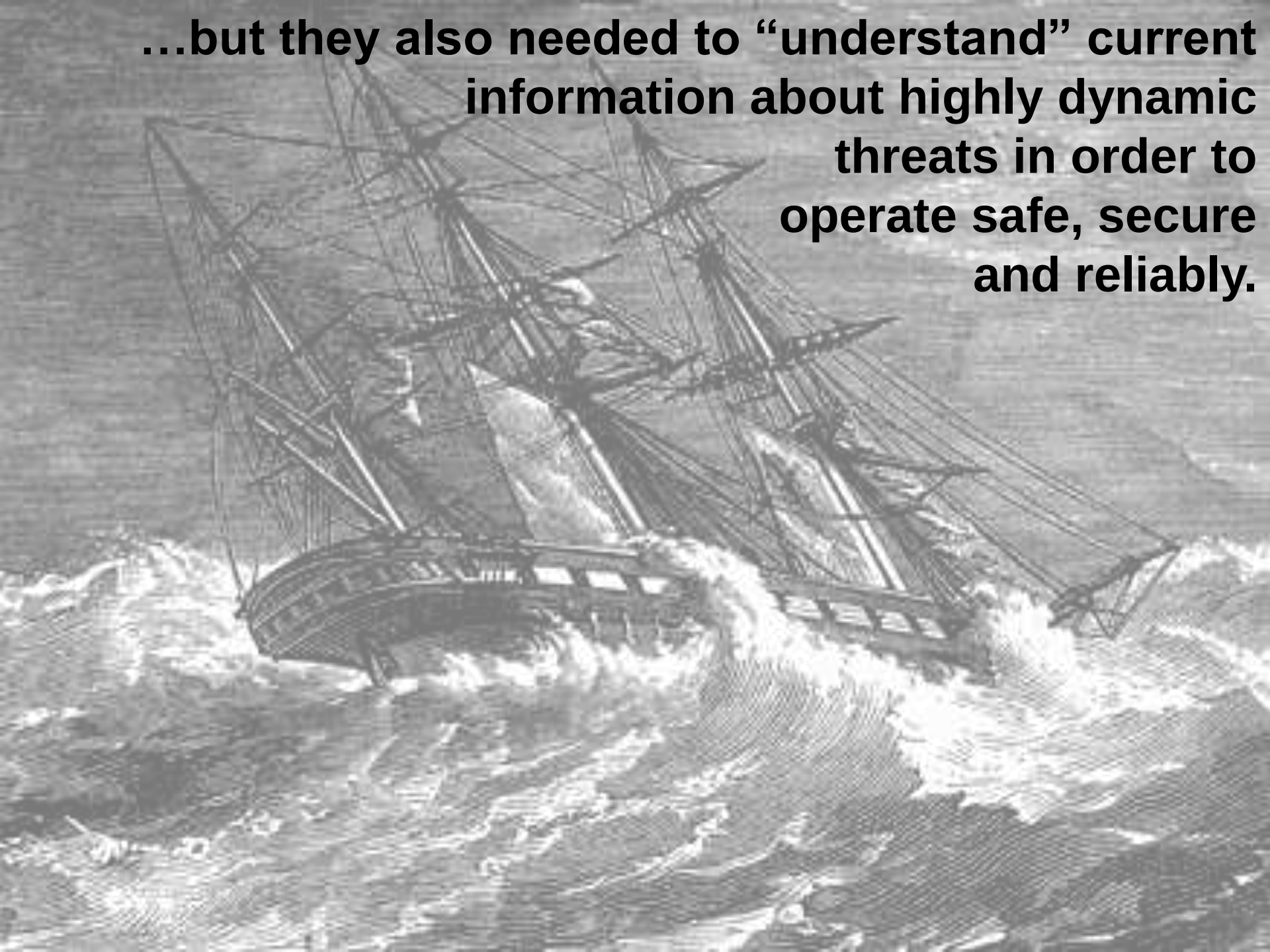

River
Know Security Weaknesses
Secure Coding Rules
Know Security Weaknesses
Know Security Weaknesses
Know Security Weaknesses
Open Waters


SEI SERIES • A CERT® BOOK
THE CERT® C SECURE CODING STANDARD
ROBERT C


MSC00-CPP. Compile cleanly at high warning levels - CERT Secure Coding Standards
https://www.securecoding.cert.org/confluence/display/cplusplus/MSC00-CPP.+Compile+cleanly+at+high+warning+levels
Software Assurance
Secure Systems
Organizational Security
Coordinated Response
Training
MSC00-CPP. Compile cleanly at high warning levels
Exceptions
Risk Assessment
References
[ISO/IEC 9899:1999] Section 5.1.1.3, "Diagnostics"
[MITRE 07] CWE ID 563, "Unused Variable"; CWE ID 570, "Expression is Always False"; CWE ID 571, "Expression is Always True"
[Sutter 05] Item 1
[Seacord 05a] Chapter 8, "Recommended Practices"
US-CERT

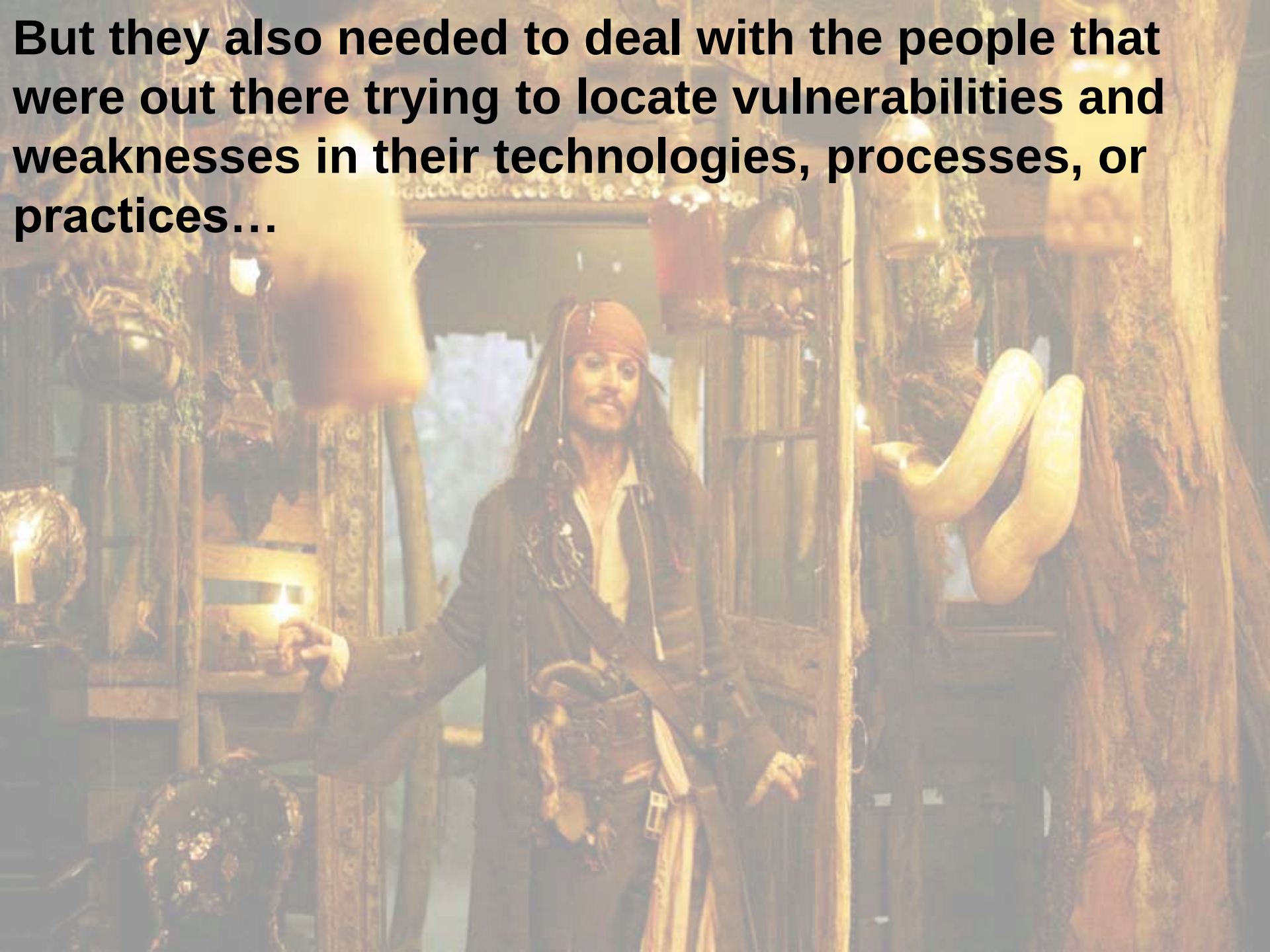
…some threats and hazards are unpredictable and dynamic…

…so new types of scanning for hazards and threats were created to make shipping safer and more dependable and secure in more places…

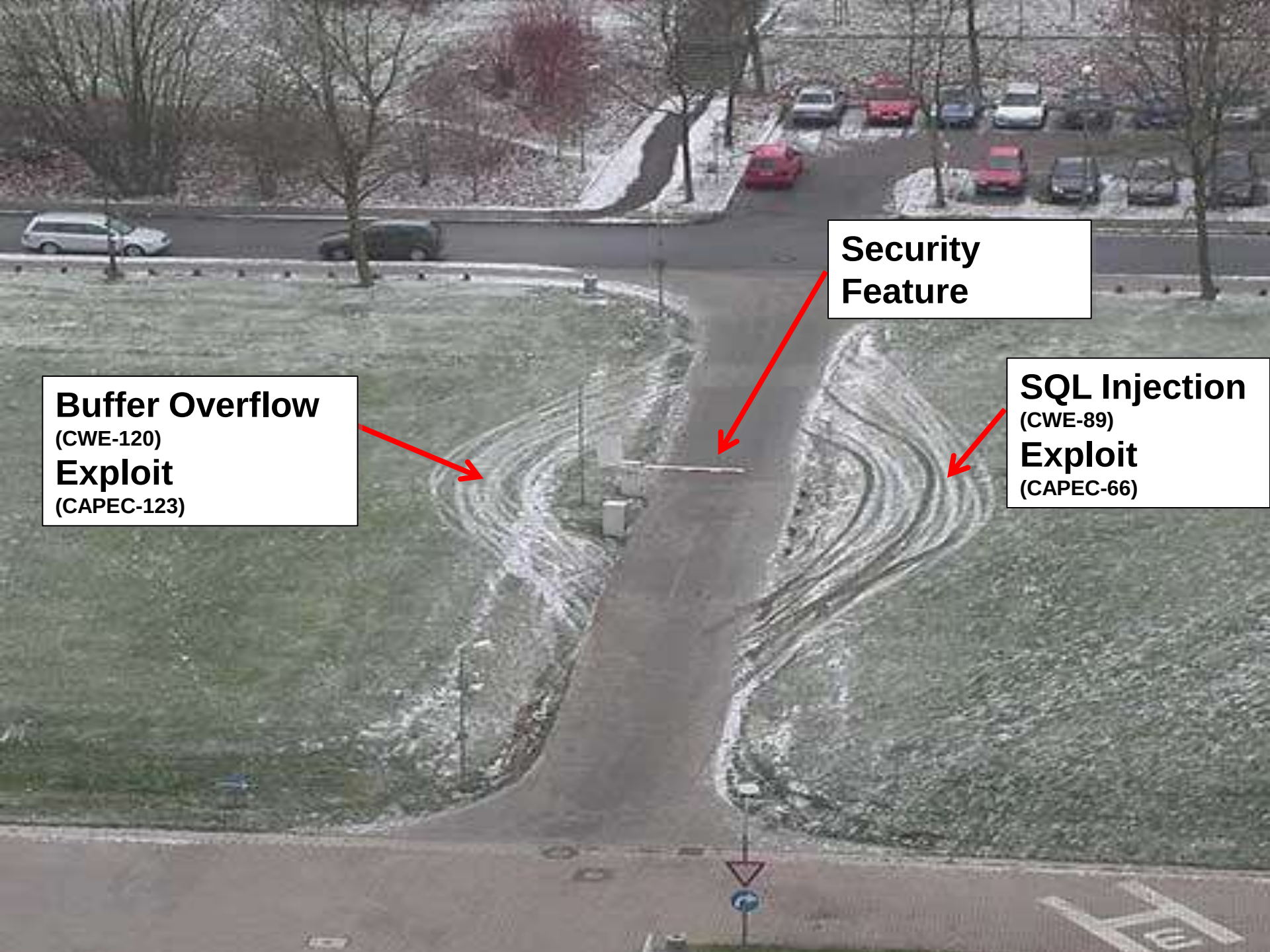


…but they also needed to “understand” current information about highly dynamic threats in order to operate safe, secure and reliably.

But they also needed to deal with the people that were out there trying to locate vulnerabilities and weaknesses in their technologies, processes, or practices…

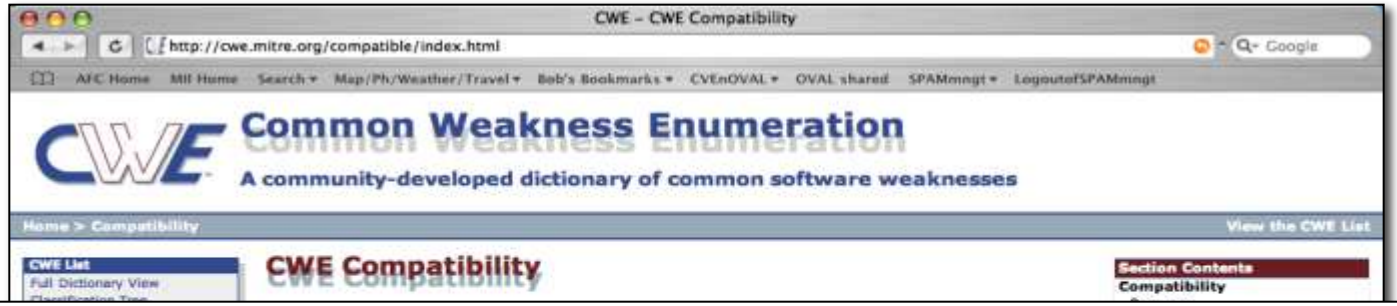



…with defensive and offensive security capabilities.
712


Security Feature
Buffer Overflow
(CWE-120)
Exploit
(CAPEC-123)
SQL Injection
(CWE-89)
Exploit
(CAPEC-66)

CWE Compatibility & Effectiveness Program

(launched Feb 2007)



SECURITY DATABASE
KEEP A VIGILANT EYE ON YOUR DEFENSES
KDM Analytics
VERACODE
FORTIFY SOFTWARE
armorize technologies
Secure Your Web Code
SecurityReason
Our Reason is Security
SkillBridge
coverity
PARASOFT
Klocwork
EMC²
CODENOMICON defensics
CENZIC
GRAMMATECH
symantec
SANS
(ISC)²
CERIAS
IPA
INFORMATION-TECHNOLOGY PROMOTION AGENCY, JAPAN
SofCheck
IBM
watchfire
checkmarx
OUNCE LABS
PRQA
Programming Research
CAST
LDRA
Software Technology
cigital
hp
SPI DYNAMICS

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.

TOTALS
Organizations Participating: **29**
Products & Services: **48**

Products are listed alphabetically by organization name:

cwe.mitre.org/compatible/

Recent Posts

MS08-078 and the SDL
Announcing CAT.NET CTP and AntiXSS v3 beta
SDL videos
BlueHat SDL Sessions Wrap-up
Secure Coding Secrets?

Tags

Common Criteria **Crawl Walk Run**
Privacy **SDL** SDL Pro Network
Security Assurance Security Blackhat
SDL **threat modeling**

News

Blogroll

BlueHat Security Briefings
The Microsoft Security Response Center
Michael Howard's Web Log
The Data Privacy Imperative
Security Vulnerability Research & Defense
Visual Studio Code Analysis Blog
MSRC Ecosystem Strategy Team

Books / Papers / Guidance

The Security Development Lifecycle (Howard and Lipner)
Privacy Guidelines for Developing Software Products and Services
Microsoft Security Development Lifecycle (SDL) – Portal
Microsoft Security Development Lifecycle (SDL) – Process Guidance (Web)
Microsoft Security Development Lifecycle (SDL) – Process Guidance (.doc)

MS08-078 and the SDL

Hi, Michael here.

Every bug is an opportunity to learn, and the security update that fixed the data binding bug that affected Internet Explorer users is no exception.

The Common Vulnerabilities and Exposures (CVE) entry for this bug is CVE-2008-4844.

Before I get started, I want to explain the goals of the SDL and the security work here at Microsoft. The SDL is designed as a multi-layered process to help systemically reduce security vulnerabilities; if one component of the SDL process fails to prevent or catch a bug, then some other component should prevent or catch the bug. The SDL also mandates the use of security defenses whose impact will be reflected in the "mitigations" section of a security bulletin, because we know that no software development process will catch all security bugs. As we have said many times, the goal of the SDL is to "Reduce vulnerabilities, and reduce the severity of what's missed."

In this post, I want to focus on the SDL-required code analysis, code review, fuzzing and compiler and operating system defenses and how they fared.

Background

The bug was an invalid pointer dereference in MSHTML.DLL when the code handles data binding. It's important to point out that there is no heap corruption and there is no heap-based buffer overrun!

When data binding is used, IE creates an object which contains an array of data binding objects. In the code in question, when a data binding object is released, the array length is not correctly updated leading to a function call into freed memory.

The vulnerable code looks a little like this (by the way, the real array name is _aryPXfer, but I figured ArrayOfObjectsFromIE is a little more descriptive for people not in the Internet Explorer team.)

```
int MaxIdx = ArrayOfObjectsFromIE.Size()-1;
for (int i=0; i <= MaxIdx; i++) {
    if (!ArrayOfObjectsFromIE[i])
        continue;
    ArrayOfObjectsFromIE[i]->TransferFromSource();
    ...
}
```

Here's how the vulnerability manifests itself: if there are two data transfers with the same identifier (so MaxIdx is 2), and the first transfer updates the length of the ArrayOfObjectsFromIE array when its work was done and releases its data binding object, the loop count would still be whatever MaxIdx was at the start of the loop, 2.

This is 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.

The fix was to check the maximum iteration count on each loop iteration rather than once before the loop starts; this is the correct fix for a TOCTOU bug - move the check as close as possible to the action because, in

a time-of-check-time-of-use (TOCTOU) bug that led to code calling into a freed memory block. The
on Weakness Enumeration (CWE) classification for this vulnerability is CWE-367.

TOCTOU issues. We will update our training to address this.

Our static analysis tools don't find this because the tools would need to understand the re-entrant nature of the code.

Fuzz Testing

September 2008 (5)
August 2008 (2)
July 2008 (8)
June 2008 (4)

OWASP Top Ten 2007 & 2010 use CWE refs

OWASP TOP 10

THE TEN MOST CR
APPLICATION SECU

2007 UPDATE

OWASP
The Open Web Application Security Project

OWASP Top 10 - 2010
The Ten Most Critical Web Application Security Risks

lease

Our methodology for the Top 10 2007 was simple: take the MITRE Vulnerability Trends for 2006, and distill the Top 10 *web application security* issues. The ranked results are as follows:

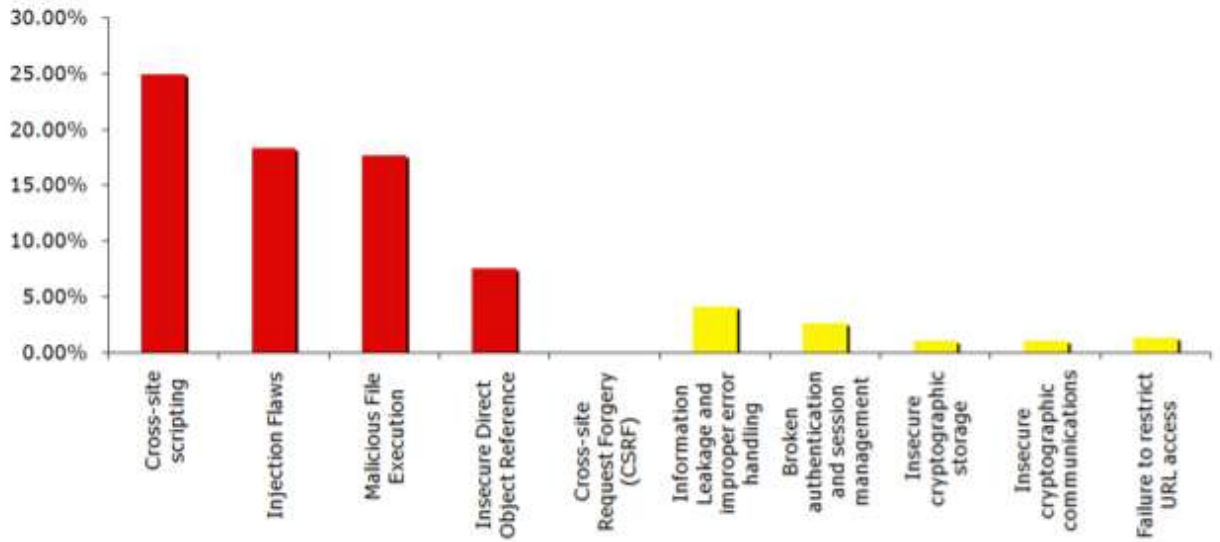


Figure 2: MITRE data on Top 10 web application vulnerabilities for 2006

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
Add

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/navlist.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: MILW0RM (disclaimer)
Name: 4443
Hyperlink: http://www.milw0rm.com/exploits/4443

Vulnerable software and versions

Configuration 1
– Clansphere, Clansphere, 2007.4

Technical Details

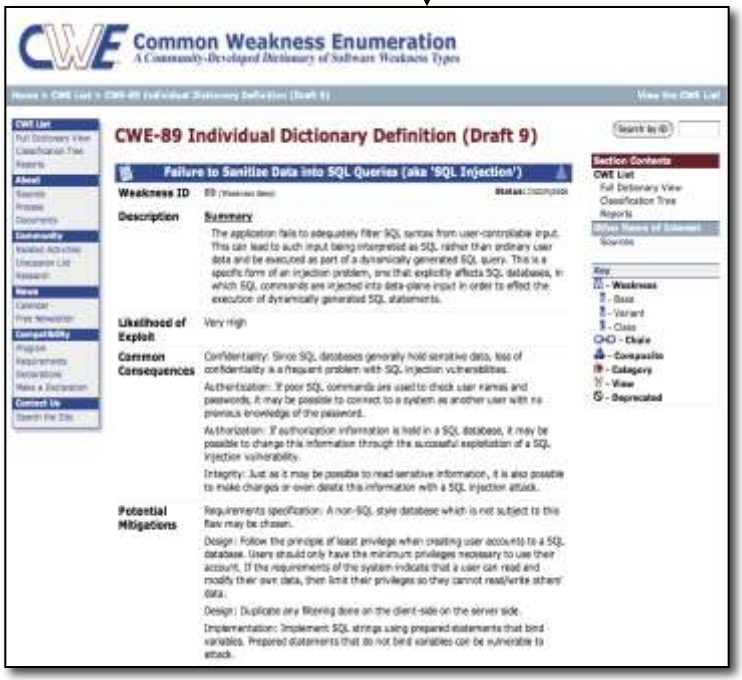
Vulnerability Type (View All)
SQL Injection (CWE-89)

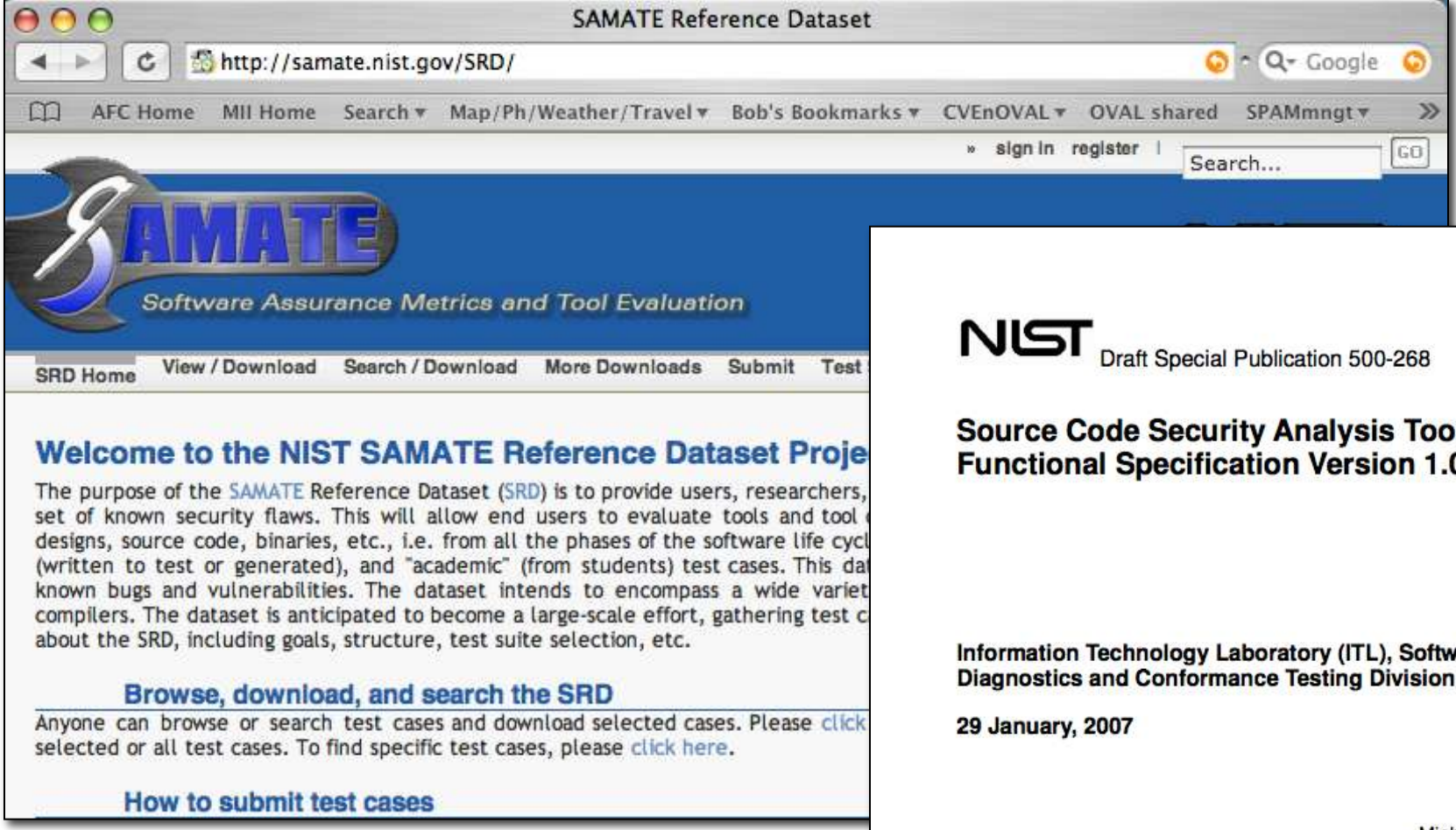
CVE Standard Vulnerability Entry:
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-5061

Common Platform Enumeration:

NVD XML feeds also include CWE

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





NIST Draft Special Publication 500-268

Source Code Security Analysis Tool Functional Specification Version 1.0

Information Technology Laboratory (ITL), Software Diagnostics and Conformance Testing Division

29 January, 2007

Michael Kass
Michael Koo

National Institute of Standards and Technology
Information Technology Laboratory
Software Diagnostics and Conformance Testing Division

NIST Special Publications:

SP800-36	CVE
SP800-40	CVE, OVAL
SP800-42	CVE
SP800-44	CVE
SP800-51	CVE
SP800-53a	CVE, OVAL, CWE
SP800-61	CVE, OVAL
SP800-70	CVE, OVAL, CCE, CPE, XCCDF, CVSS
SP800-82	CVE
SP800-86	CVE
SP800-94	CVE
SP800-115	CVE, CCE, CVSS, CWE
SP800-117	CVE, OVAL, CCE, CPE, XCCDF, CVSS
SP800-126	CVE, OVAL, CCE, CPE, XCCDF, CVSS

NIST Interagency Reports:

NISTIR-7007	CVE
NISTIR-7275	CVE, OVAL, CCE, CPE, XCCDF, CVSS
NISTIR-7435	CVE, CVSS, CWE
NISTIR-7511	CVE, OVAL, CCE, CPE, XCCDF, CVSS
NISTIR-7517	CVE
NISTIR-7581	CVE
NISTIR-7628	CVE, CWE

FDCC

SCAP™

EMAP

SwAAP

Industry Uptake

Manually review code after security education

Manual code review, especially review of high-risk code, such as code that faces the Internet or parses data from the Internet, is critical, but only if the people performing the code review know what to look for and how to fix any code vulnerabilities they find. The best way to help understand classes of security bugs and remedies is education, which should minimally include the following areas:

- C and C++ vulnerabilities and remedies, most notably buffer overruns and integer arithmetic issues.
- Web-specific vulnerabilities and remedies, such as cross-site scripting (XSS).
- Database-specific vulnerabilities and remedies, such as SQL injection.
- Common cryptographic errors and remedies.

Many vulnerabilities are programming language (C, C++ etc) or domain-specific (web, database) and others can be categorized by vulnerability type, such as injection (XSS and SQL Injection) or cryptographic (poor random number generation and weak secret storage) so specific training in these areas is advised.

Resources

- A Process for Performing Security Code Reviews, Michael Howard, IEEE Security & Privacy July/August 2006.
- .NET Framework Security — Code Review;
- Common Weakness Enumeration, MITRE; http://cwe.mitre.org/
- Security Code Reviews; http://www.codesecurely.org/Wiki/view.aspx/Security_Code_Reviews
- Security Code Review — Use Visual Studio Bookmarks To Capture Security Findings; http://blogs.msdn.com/alikl/archive/2008/01/24/security-code-review-use-visual-studio-bookmarks-to-capture-security-findings.aspx
- Security Code Review Guidelines, Adam Shostack; http://www.verber.com/mark/cs/security/code-review.html
- OWASP Top Ten; http://www.owasp.org/index.php/OWASP_Top_Ten_Project

Testing

Testing activities validate the secure implementation of a product, which re
the likelihood of security bugs being released and discovered by customers a
malicious users. The majority of SAFECode members have adopted the fully
software security testing practices in their software development lifecycle. Th
is not to "test in security," but rather to validate the robustness and secur
the software products prior to making the product available to customers. T
testing methods do find security bugs, especially for products that may not
undergone critical secure development process changes.

Fuzz testing

Fuzz testing is a reliability and security testing technique that relies on bu
intentionally malformed data and then having the software under test consume the
malformed data to see how it responds. The science of fuzz testing is somewhat
new but it is maturing rapidly. There is a small market for fuzz testing tools today,
but in many cases software developers must build bespoke fuzz testers to suit spe-
cialized file and network data formats. Fuzz testing is an effective testing technique
because it uncovers weaknesses in data handling code.

Resources

- Fuzz Testing of Application Reliability, University of Wisconsin; http://pages.cs.wisc.edu/~bart/fuzz/fuzz.html
- Automated Whitebox Fuzz Testing, Michael Levin, Patrice Godefroid and Dave Molnar, Microsoft Research; ftp://ftp.research.microsoft.com/pub/tr/TR-2007-58.pdf
- IANewsletter Spring 2007 "Look out! It's the fuzz!" Matt Warnock; http://iac.dtic.mil/iatac/download/Vol10_No1.pdf
- *Fuzzing: Brute Force Vulnerability Discovery*, Sutton, Greene & Amini, Addison-Wesley
- Open Source Security Testing Methodology Manual, ISECOM
- *Common Attack Pattern Enumeration and Classification*, MITRE; http://capec.mitre.org/

SAFECode
Driving Security and Integrity

SAFECode
Driving Security and Integrity

Fundamental Practices for Secure Software Development

A Guide to the Most Effective Secure Development Practices in Use Today

OCTOBER 8, 2008

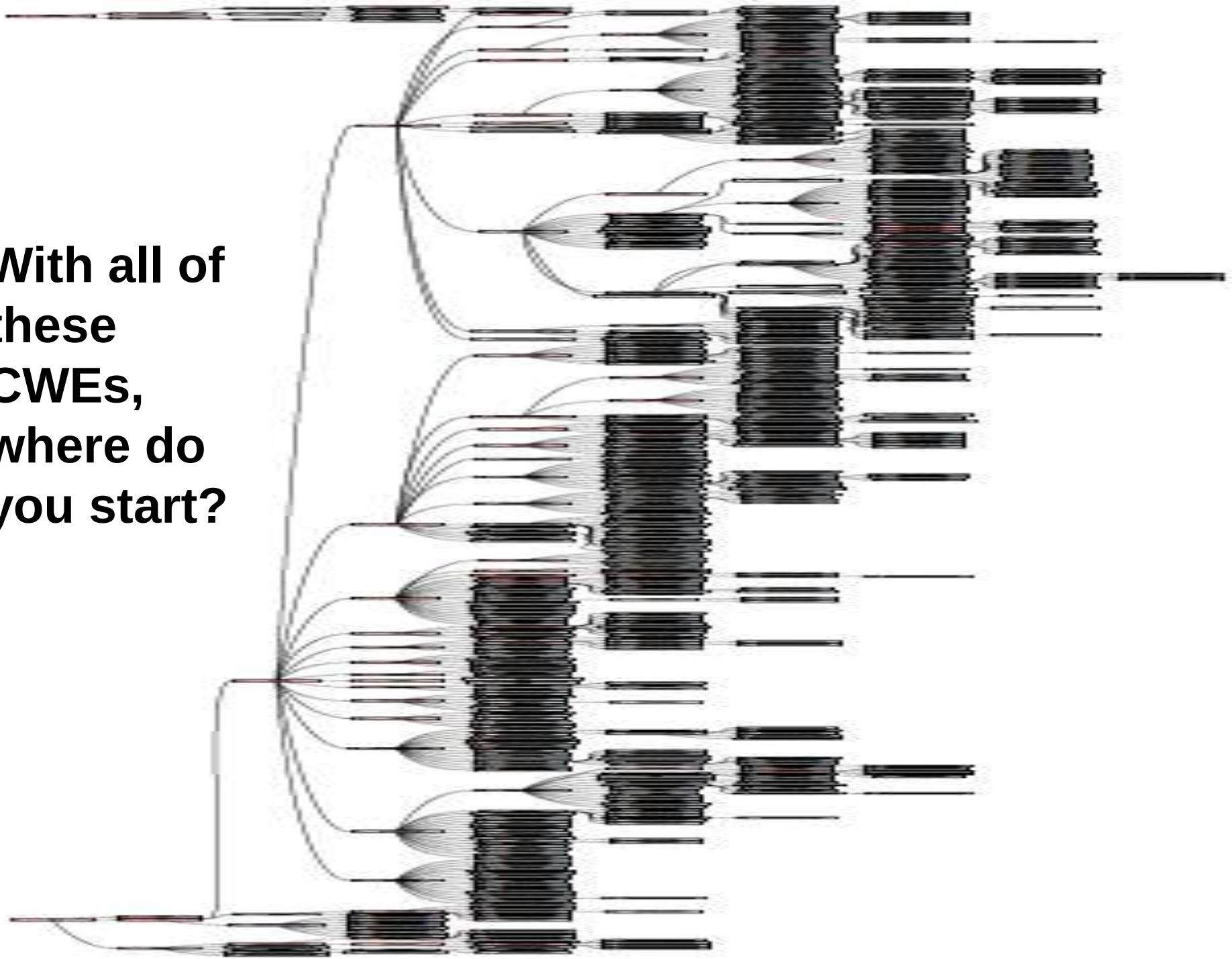
Lead Writer: Michael Howard, Microsoft Corp.

Contributors

Gunter Bitz, SAP AG
Jerry Cochran, Microsoft Corp.
Matt Coles, EMC Corporation
Danny Dhillon, EMC Corporation
Chris Fagan, Microsoft Corp.
Cassio Goldschmidt, Symantec Corp.
Wesley Higaki, Symantec Corp.
Steve Lipner, Microsoft Corp.
Brad Minnis, Juniper Networks, Inc.
Handik Parekh, EMC Corporation
Dan Reddy, EMC Corporation
Alexandr Seleznyov, Nokia
Reany Sondhi, EMC Corporation
Janne Uusilehto, Nokia
Antti Vähä-Sipilä, Nokia

CWE
CAPEC

With all of these CWEs, where do you start?



20010 CWE/SANS Top 25 Programming Errors
(released 16 Feb 2010) cwe.mitre.org/top25/

- **Sponsored by:**
  - National Cyber Security Division (DHS)
- **List was selected by a group of security experts from 34 organizations including:**
  - Academia: Purdue, Northern Kentucky University
  - Government: CERT, NSA, DHS
  - Software Vendors: Microsoft, Oracle, Red Hat, Apple, Juniper, McAfee, Symantec, Sun, RSA (of EMC)
  - Security Vendors: Veracode, Fortify, Cigital, Mandiant, Cigital, Secunia, Breach, SAIC, Aspect, WhiteHat
  - Security Groups: OWASP, WASC


Robert C. Seacord
CERT
Ryan Barnett
Pascal Meunier
CERIAS, Purdue University
Antonio Fontes
Matt Bishop
Kenneth van Wyk
Masato Terada
Sean Barnum
Mahesh Saptarshi
Cassio Goldschmidt
Adam Hahn
Jeff Williams
Carsten Eiram
Josh Drake
Chuck Willis
Michael Howard
Bruce Lowenthal
Mark J. Cox
Jacob West
Djenana Campara
James Walden
Frank Kim
Chris Eng
Chris Wysopal
Veracode, Inc.
2009
CWE - Top 25 Credited Contributors
http://cwe.mitre.org/top25/contributors.html
Common Weakness Enumeration
A Community-Developed Dictionary of Software Weakness Types
2010
CERIAS
Center for Education and Research in Information Assurance and Security
Microsoft
ORACLE
cigital
Sun microsystems
EMC²
redhat
Secunia
Stay Secure
SRI International
NSA
DHS
IPA
INFORMATION-TECHNOLOGY PROMOTION AGENCY, JAPAN
SANS
GRAMMATECH
SRA
FORTIFY SOFTWARE
symantec.
armorize technologies
Secure Your Web Code
VERACODE
McAfee
Juniper NETWORKS
Section Contents
CWE/SANS Top 25
Contributors
Supporting Quotes
Monster Mitigations
Focus Profiles
On the Cusp
Documents & Podcasts
Training Materials
Top 25 FAQ
Top 25 Process
Change Log
SANS News Release
Section Archives
2009 CWE/SANS Top 25
Supporting Quotes
Contributors
On The Cusp
Change Log

Insecure Interaction Between Components

These weaknesses are related to insecure ways in which data is sent and received between separate components, modules, programs, processes, threads, or systems.

For each weakness, its ranking in the general list is provided in square brackets.

Rank	CWE ID	Name
[1]	CWE-79	Failure to Preserve Web Page Structure ('Cross-site Scripting')
[2]	CWE-89	Improper Sanitization of Special Elements used in an SQL Command ('SQL Injection')
[4]	CWE-352	Cross-Site Request Forgery (CSRF)
[8]	CWE-434	Unrestricted Upload of File with Dangerous Type
[9]	CWE-78	Improper Sanitization of Special Elements used in an OS Command ('OS Command Injection')
[17]	CWE-209	Information Exposure Through an Error Message
[23]	CWE-601	URL Redirection to Untrusted Site ('Open Redirect')
[25]	CWE-362	Race Condition

Risky Resource Management

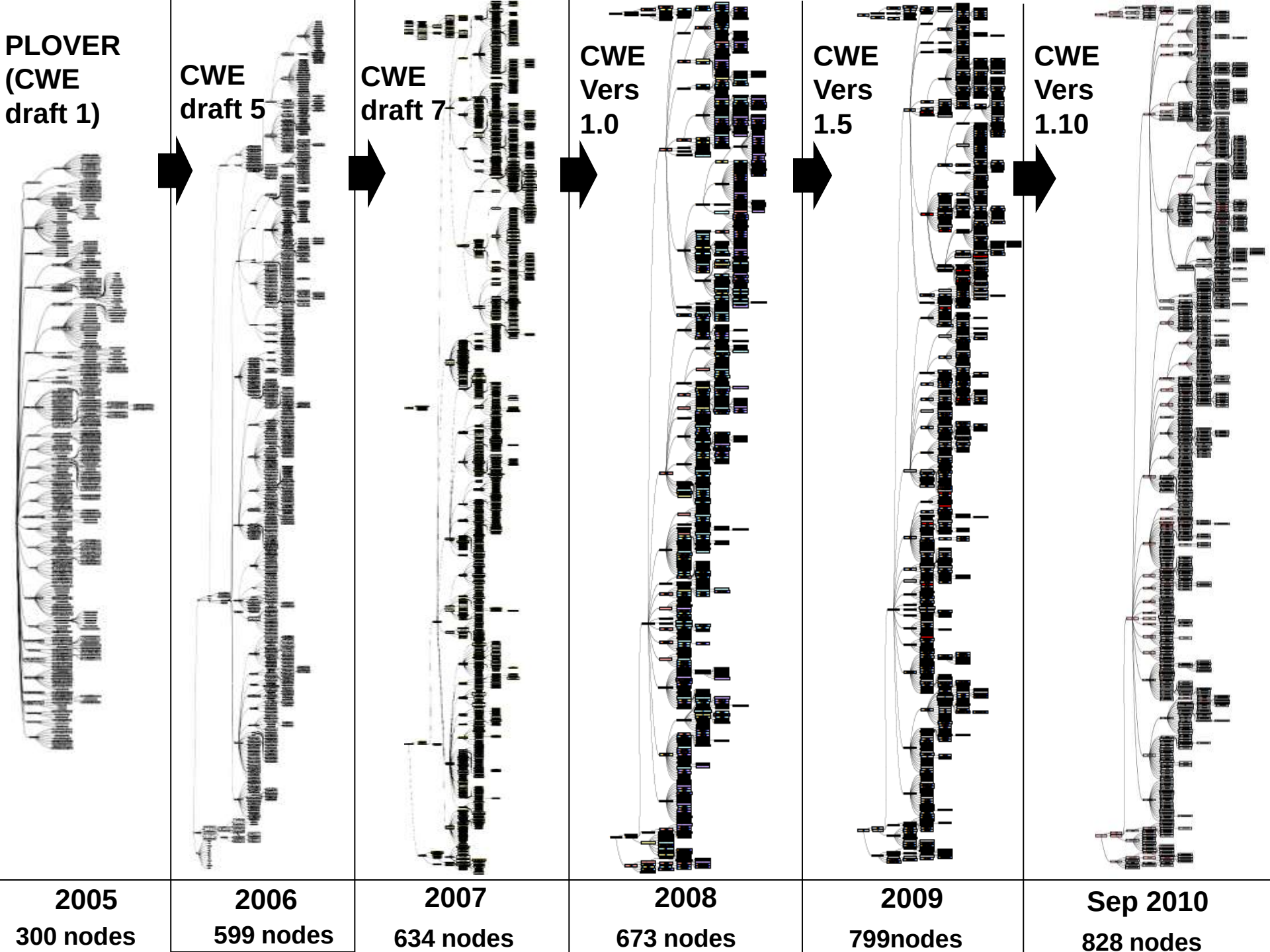
The weaknesses in this category are related to ways in which software does not properly manage the creation, usage, transfer, or destruction of important system resources.

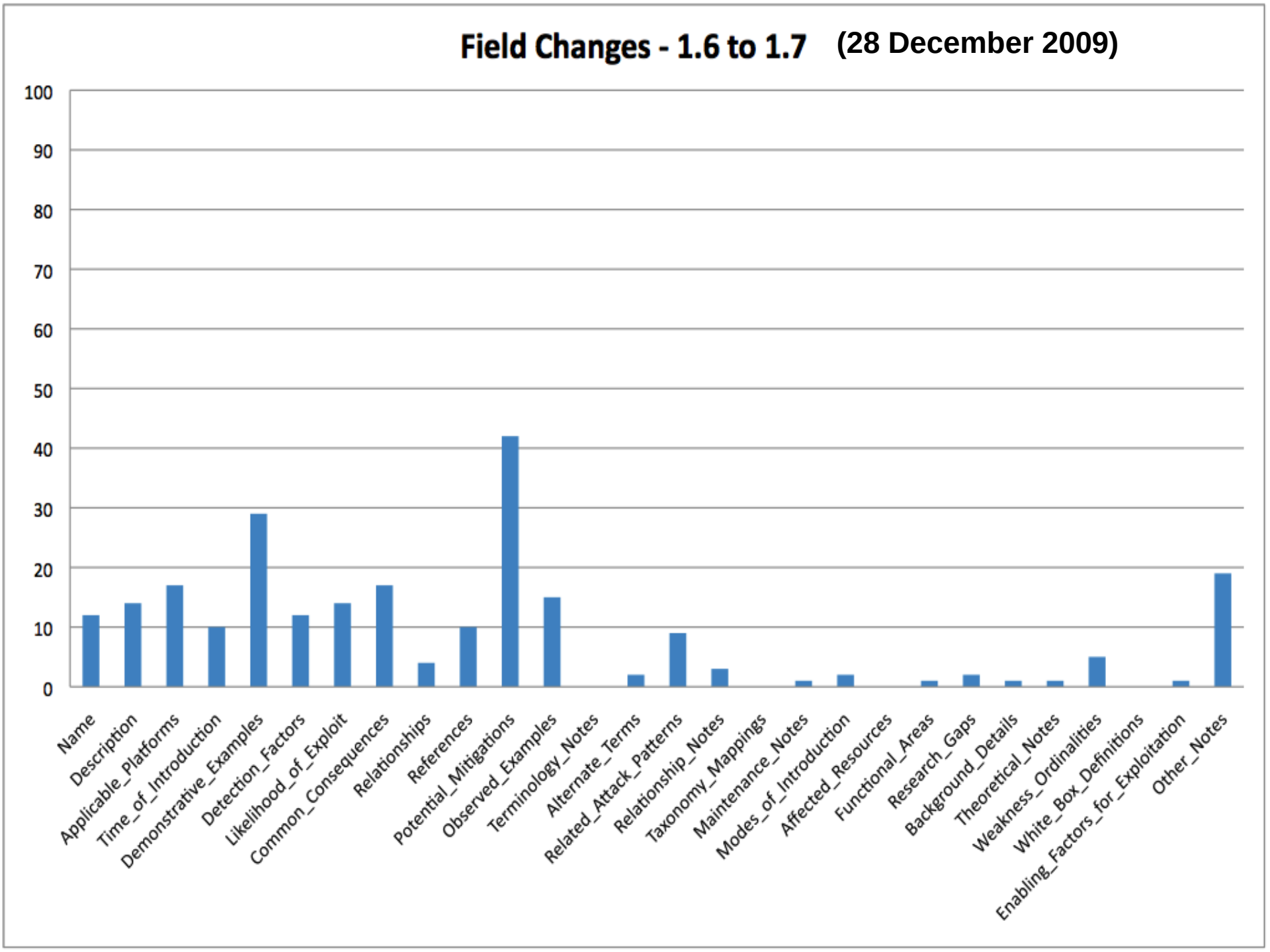
Rank	CWE ID	Name
[3]	CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
[7]	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
[12]	CWE-805	Buffer Access with Incorrect Length Value
[13]	CWE-754	Improper Check for Unusual or Exceptional Conditions
[14]	CWE-98	Improper Control of Filename for Include/Require Statement in PHP Program ('PHP File Inclusion')
[15]	CWE-129	Improper Validation of Array Index
[16]	CWE-190	Integer Overflow or Wraparound
[18]	CWE-131	Incorrect Calculation of Buffer Size
[20]	CWE-494	Download of Code Without Integrity Check
[22]	CWE-770	Allocation of Resources Without Limits or Throttling

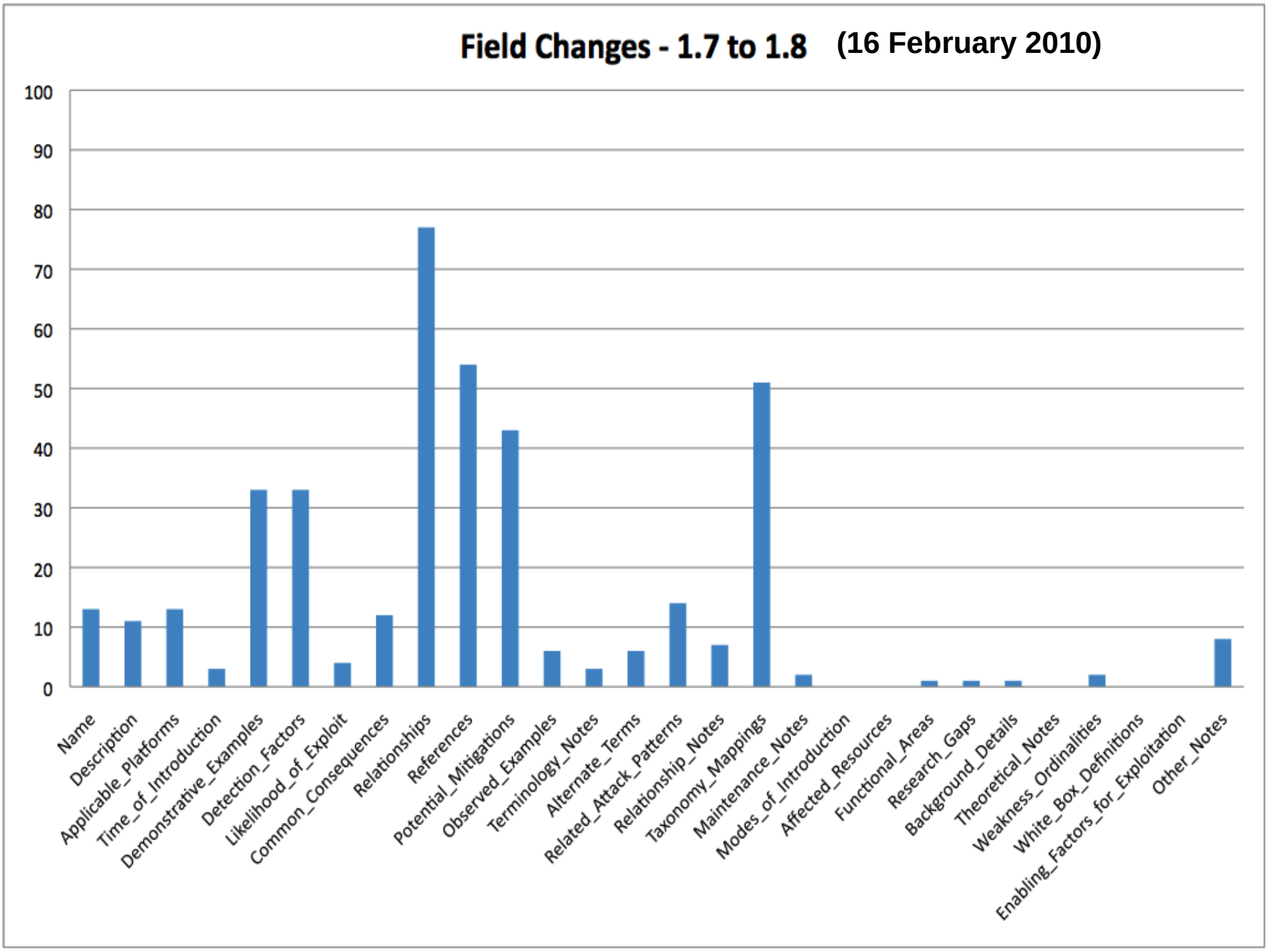
Porous Defenses

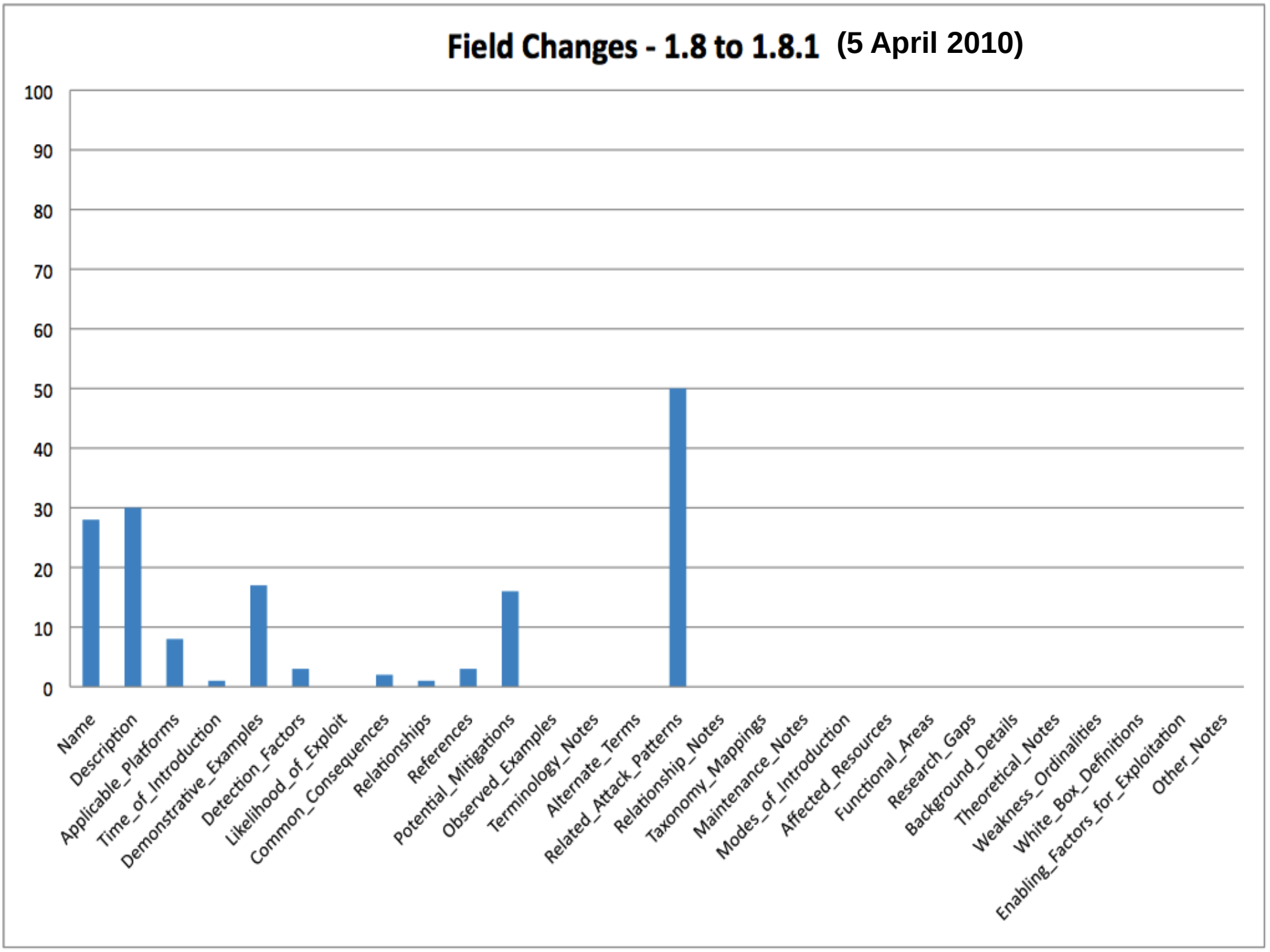
The weaknesses in this category are related to defensive techniques that are often misused, abused, or just plain ignored.

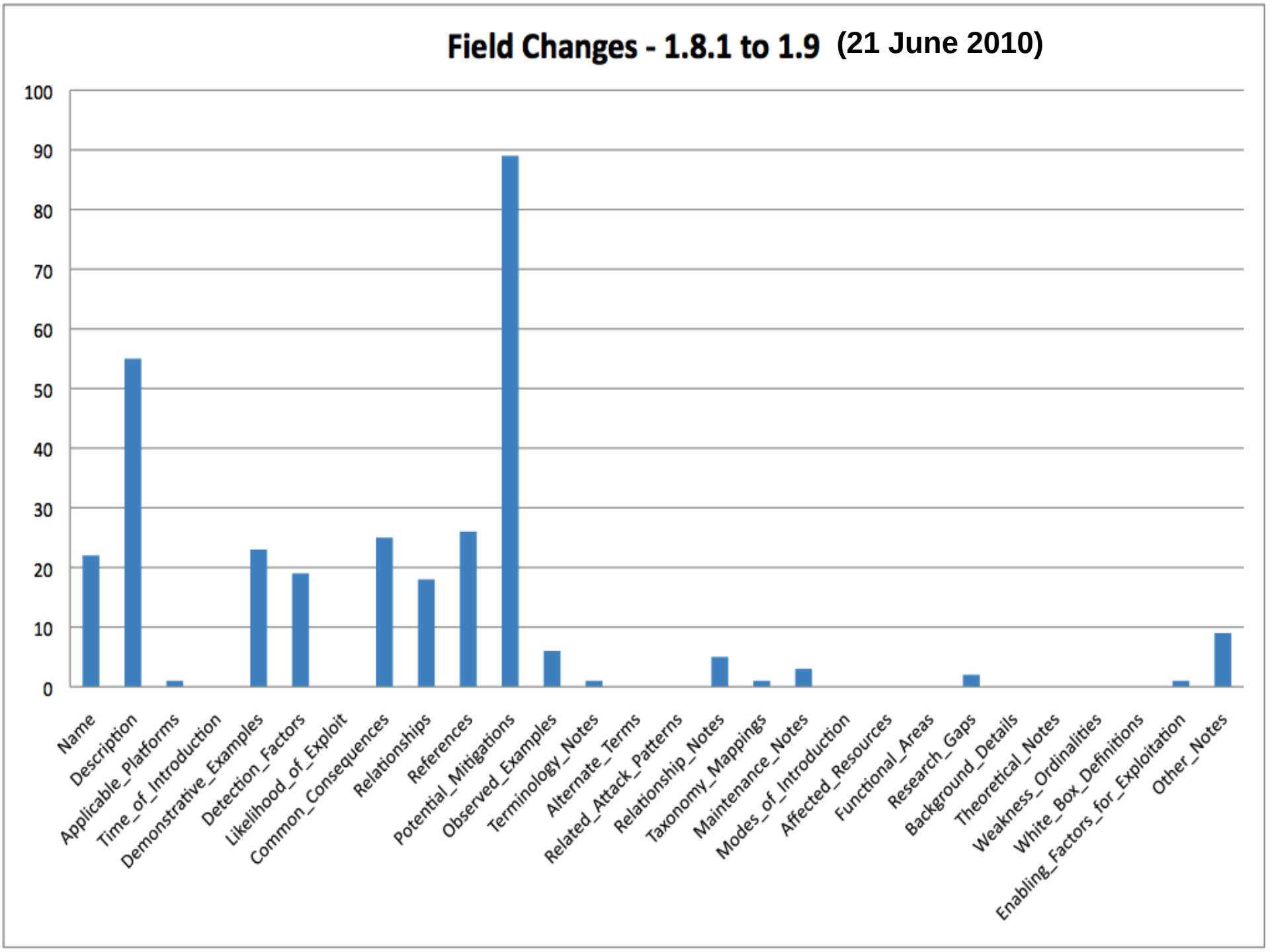
Rank	CWE ID	Name
[5]	CWE-285	Improper Access Control (Authorization)
[6]	CWE-807	Reliance on Untrusted Inputs in a Security Decision
[10]	CWE-311	Missing Encryption of Sensitive Data
[11]	CWE-798	Use of Hard-coded Credentials
[19]	CWE-306	Missing Authentication for Critical Function
[21]	CWE-732	Incorrect Permission Assignment for Critical Resource
[24]	CWE-327	Use of a Broken or Risky Cryptographic Algorithm

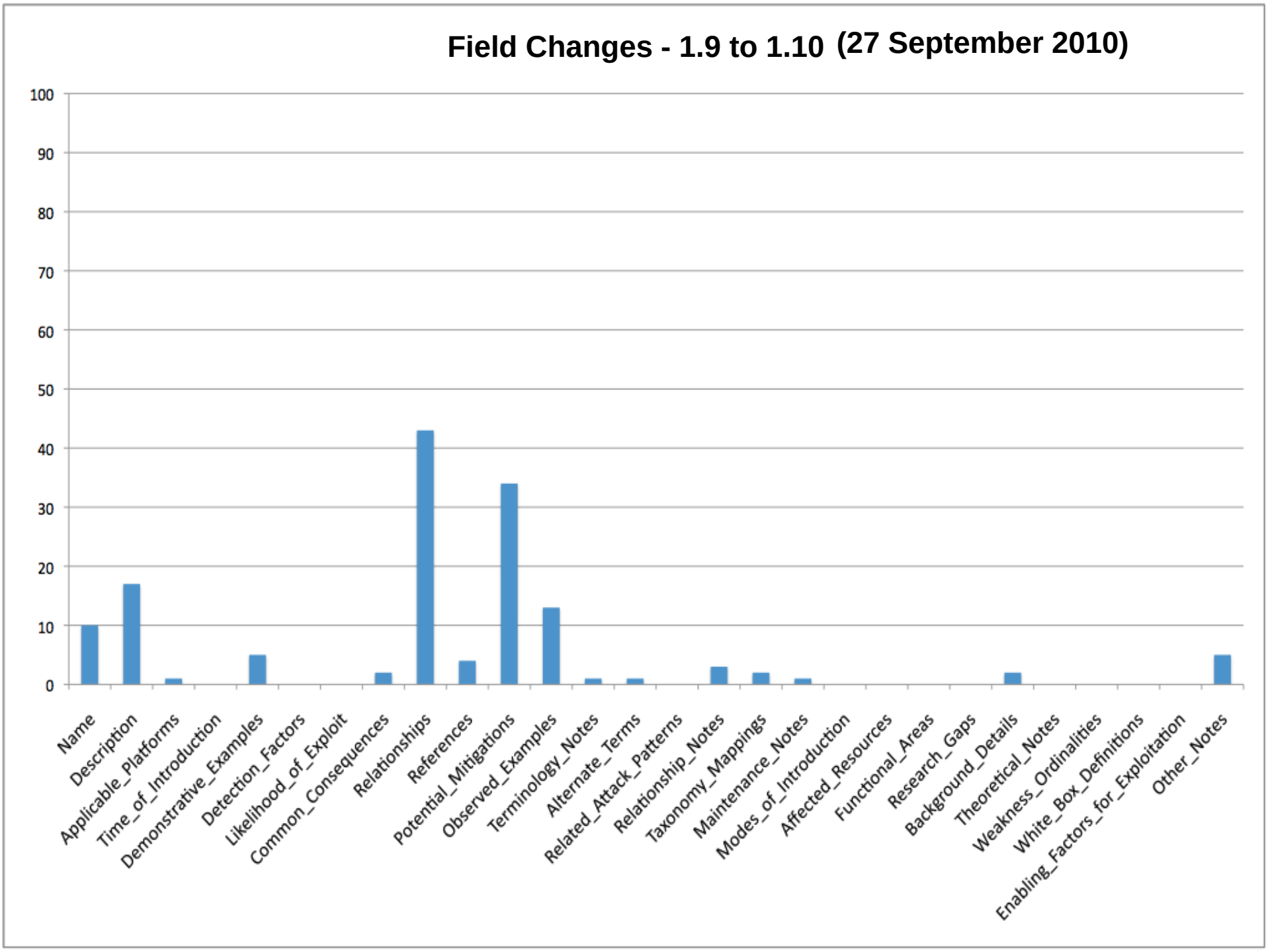

PLOVER (CWE draft 1)
CWE draft 5
CWE draft 7
CWE Vers 1.0
CWE Vers 1.5
CWE Vers 1.10
2005
300 nodes
2006
599 nodes
2007
634 nodes
2008
673 nodes
2009
799nodes
Sep 2010
828 nodes


Field Changes - 1.6 to 1.7 (28 December 2009)
100
90
80
70
60
50
40
30
20
10
0
Name
Description
Applicable_Platforms
Time_of_Introduction
Demonstrative_Examples
Detection_Factors
Likelihood_of_Exploit
Common_Consequences
Relationships
References
Potential_Mitigations
Observed_Examples
Terminology_Notes
Alternate_Terms
Related_Attack_Patterns
Relationship_Notes
Taxonomy_Mappings
Maintenance_Notes
Modes_of_Introduction
Affected_Resources
Functional_Areas
Research_Gaps
Background_Details
Theoretical_Notes
Weakness_Ordinalities
White_Box_Definitions
Enabling_Factors_for_Exploitation
Other_Notes


Field Changes - 1.7 to 1.8 (16 February 2010)
100
90
80
70
60
50
40
30
20
10
0
Name
Description
Applicable_Platforms
Time_of_Introduction
Demonstrative_Examples
Detection_Factors
Likelihood_of_Exploit
Common_Consequences
Relationships
References
Potential_Mitigations
Observed_Examples
Terminology_Notes
Alternate_Terms
Related_Attack_Patterns
Relationship_Notes
Taxonomy_Mappings
Maintenance_Notes
Modes_of_Introduction
Affected_Resources
Functional_Areas
Research_Gaps
Background_Details
Theoretical_Notes
Weakness_Ordinalities
White_Box_Definitions
Enabling_Factors_for_Exploitation
Other_Notes


Field Changes - 1.8 to 1.8.1 (5 April 2010)
100
90
80
70
60
50
40
30
20
10
0
Name
Description
Applicable_Platforms
Time_of_Introduction
Demonstrative_Examples
Detection_Factors
Likelihood_of_Exploit
Common_Consequences
Relationships
References
Potential_Mitigations
Observed_Examples
Terminology_Notes
Alternate_Terms
Related_Attack_Patterns
Relationship_Notes
Taxonomy_Mappings
Maintenance_Notes
Modes_of_Introduction
Affected_Resources
Functional_Areas
Research_Gaps
Background_Details
Theoretical_Notes
Weakness_Ordinalities
White_Box_Definitions
Enabling_Factors_for_Exploitation
Other_Notes


Field Changes - 1.8.1 to 1.9 (21 June 2010)
100
90
80
70
60
50
40
30
20
10
0
Name
Description
Applicable_Platforms
Time_of_Introduction
Demonstrative_Examples
Detection_Factors
Likelihood_of_Exploit
Common_Consequences
Relationships
References
Potential_Mitigations
Observed_Examples
Terminology_Notes
Alternate_Terms
Related_Attack_Patterns
Relationship_Notes
Taxonomy_Mappings
Maintenance_Notes
Modes_of_Introduction
Affected_Resources
Functional_Areas
Research_Gaps
Background_Details
Theoretical_Notes
Weakness_Ordinalities
White_Box_Definitions
Enabling_Factors_for_Exploitation
Other_Notes


Field Changes - 1.9 to 1.10 (27 September 2010)
100
90
80
70
60
50
40
30
20
10
0
Name
Description
Applicable_Platforms
Time_of_Introduction
Demonstrative_Examples
Detection_Factors
Likelihood_of_Exploit
Common_Consequences
Relationships
References
Potential_Mitigations
Observed_Examples
Terminology_Notes
Alternate_Terms
Related_Attack_Patterns
Relationship_Notes
Taxonomy_Mappings
Maintenance_Notes
Modes_of_Introduction
Affected_Resources
Functional_Areas
Research_Gaps
Background_Details
Theoretical_Notes
Weakness_Ordinalities
White_Box_Definitions
Enabling_Factors_for_Exploitation
Other_Notes

CWE version 1.10

HS SEDI
Homeland Security Systems Engineering and Development Institute

Summary of Entry Types

Type	Version 1.9	v1.10
Category	119	119
Chain	3	3
Composite	6	6
Deprecated	11	11
View	24	24
Weakness	658	665

Nodes Added to v1.10

CWE-ID	CWE Name
820	Missing Synchronization
821	Incorrect Synchronization
822	Untrusted Pointer Dereference
823	Use of Out-of-range Pointer Offset
824	Access of Uninitialized Pointer
825	Expired Pointer Dereference
826	Premature Release of Resource During Expected Lifetime

CWE web site visitors by City

Recent Posts

SDL Threat Modeling Tool 3.1.4 ships!
Early Days of the SDL, Part Four
Early Days of the SDL, Part Three
Early Days of the SDL, Part Two
Early Days of the SDL, Part One

Tags

Common Criteria Crawl Walk Run Privacy SDL SDL Pro Network Security Assurance Security Blackhat SDL threat modeling

News

About Us

Adam Shostack
Bryan Sullivan
David Ladd
Jeremy Dallman
Michael Howard
Steve Lipner

Blogroll

BlueHat Security Briefings

SDL and the CWE/SANS Top 25

Bryan here. The security community has been buzzing since SANS and MITRE's joint announcement earlier this month of their list of the Top 25 Most Dangerous Programming Errors. Now, I don't want to get into a debate in this blog about whether this new list will become the new de facto standard for analyzing security vulnerabilities (or indeed, whether it already has become the new standard). Instead, I'd like to present an overview of how the Microsoft SDL maps to the CWE/SANS list, just
May.

Michael and I have writte
coverage of the Top 25 ar
believe that the results te
25 were developed indep
root them out of the softw
analysis white paper and
guidance around every m
made many of the same S
for you to download and u

Below is a summary of ho
see the SDL covers every
them (race conditions and
by multiple SDL requirem
tools to prevent or detect

CWE	Title
20	Improper Input Va
116	Improper Encodin Escaping of Outpu

CWE	Title	Education?	Manual Process?	Tools?	Threat Model?
20	Improper Input Validation	Y	Y	Y	Y
116	Improper Encoding or Escaping of Output	Y	Y	Y	
89	Failure to Preserve SQL Query Structure (aka SQL Injection)	Y	Y	Y	
79	Failure to Preserve Web Page Structure (aka Cross-Site Scripting)	Y	Y	Y	
78	Failure to Preserve OS Command Structure (aka OS Command Injection)	Y		Y	
319	Cleartext Transmission of Sensitive Information	Y			Y
352	Cross-site Request Forgery (aka CSRF)	Y		Y	
362	Race Condition	Y			
209	Error Message Information Leak	Y	Y	Y	
119	Failure to Constrain Memory Operations within the Bounds of a Memory Buffer	Y	Y	Y	
642	External Control of Critical State Data	Y			Y
73	External Control of File Name or Path	Y	Y	Y	
426	Untrusted Search Path	Y		Y	
94	Failure to Control Generation of Code (aka 'Code Injection')	Y	Y		
494	Download of Code Without Integrity Check				Y
404	Improper Resource Shutdown or Release	Y		Y	
665	Improper Initialization	Y		Y	
682	Incorrect Calculation	Y		Y	
285	Improper Access Control (Authorization)	Y	Y		Y
327	Use of a Broken or Risky Cryptographic Algorithm	Y	Y	Y	
259	Hard-Coded Password	Y	Y	Y	Y
732	Insecure Permission Assignment for Critical Resource	Y	Y		
330	Use of Insufficiently Random Values	Y	Y	Y	
250	Execution with Unnecessary Privileges	Y	Y		Y
602	Client-Side Enforcement of Server-Side Security	Y			Y

CWE Outreach: A Team Sport

May/June Issue of IEEE Security & Privacy…

SANS Common Security Errors in Programming

Special thanks to Robert A. Martin of MITRE Corporation.

The SANS Common Security Errors in Programming map illustrates the software weaknesses that are responsible for the majority of the publicly known vulnerabilities discovered in 2006. It is based on the CWE (Common Weakness Enumeration) that provides a unified, measurable set of software weaknesses that will enable more effective discussion and action to find these weaknesses in source code and eliminate them. The CWE was developed by MITRE and sponsored by the Department of Homeland Security. The numbers between parentheses represent the common weakness enumeration IDs for each weakness. Numbers between square brackets are direct children of the weakness listed. CWE IDs can be found at the MITRE CWE Website or accessed directly by putting the number (in place of ###) in the following URL:

http://cwe.mitre.org/data/definitions/###.html

SANS MITRE

2009's Top 25 CVE Causes and Important CWEs

Handler Errors

- Deployment of Wrong Handler
- Missing Handler
- Dangerous Handler not Disabled During Sensitive Operations
- Unparsed Raw Web Content Delivery
- Incomplete Identification of Uploaded File Variables (PHP)
- Unrestricted File Upload

User Interface Errors

- UI Discrepancy for Security Feature
- Multiple Interpretations of UI Input
- UI Misrepresentation of Critical Information

Behavioral Problems

- Behavioral Change in New Version or Environment
- Expected Behavior Violation

Initialization and Cleanup Errors

- Insecure Default Variable Initialization
- External Initialization of Trusted Variables
- Non-exit on Failed Initialization
- Missing Initialization
- Incomplete Cleanup
- Improper Cleanup on Thrown Exception
- Improper Initialization - (665)

Channel and Path Errors

- Channel Errors
- Failure to Protect Alternate Path
- Uncontrolled Search Path Element
- Unquoted Search Path or Element
- Untrusted Search Path

Error Handling

- Error Conditions, Return Values, Status Codes
- Failure to Use a Standardized Error Handling Mechanism
- Failure to Catch All Exceptions in Servlet
- Not Failing Securely ('Failing Open')
- Missing Custom Error Page

Pointer Issues

- Return of Pointer Value Outside of Expected Range
- Use of size of() on a Pointer Type
- Incorrect Pointer Scaling
- Use of Pointer Subtraction to Determine Size
- Assignment of a Fixed Address to a Pointer
- Attempt to Access Child of a Non-structure Pointer

Time and State

- State Issues
  - Incomplete Internal State Distinction
  - State Synchronization Error
  - Mutable Objects Passed by Reference
  - Passing Mutable Objects to an Untrusted Method
  - External Control of Critical State Data - (642)
- Race Condition - (362)
- Session Fixation
- Concurrency Issues
- Temporary File Issues
- Covert Timing Channel
- Technology-Specific Time and State Issues
- Symbolic Name not Mapping to Correct Object
- Signal Errors
- Unrestricted Externally Accessible Lock
- Double-Checked Locking
- Insufficient Session Expiration
- Insufficient Synchronization
- Use of a Non-reentrant Function in an Unsynchronized Context
- Improper Control of a Resource Through its Lifetime
- Exposure of Resource to Wrong Sphere
- Incorrect Resource Transfer Between Spheres
- Use of a Resource after Expiration or Release
- External Influence of Sphere Definition
- Uncontrolled Recursion
- Redirect Without Exit

Failure to Fulfill API Contract ('API Abuse')

- Failure to Clear Heap Memory Before Release ('Heap Inspection')
- Call to Non-ubiquitous API
- Use of Inherently Dangerous Function
- Multiple Binds to the Same Port
- J2EE Bad Practices: Direct Management of Connections
- Incorrect Check of Function Return Value
- Often Misused: Arguments and Parameters
- Uncaught Exception
- Execution with Unnecessary Privileges - (250)
- Often Misused: String Management
- J2EE Bad Practices: Direct Use of Sockets
- Unchecked Return Value
- Failure to Change Working Directory in chroot Jail
- Reliance on DNS Lookups in a Security Decision
- Failure to Follow Specification
- Failure to Provide Specified Functionality

Web Problems

- Failure to Sanitize CRLF Sequences in HTTP Headers ('HTTP Response Splitting')
- Inconsistent Interpretation of HTTP Requests ('HTTP Request Smuggling')
- Improper Sanitization of HTTP Headers for Scripting Syntax
- Use of Non-Canonical URL Paths for Authorization Decisions

Indicator of Poor Code Quality

- NULL Pointer Dereference
- Incorrect Block Delimitation
- Omitted Break Statement in Switch
- Undefined Behavior for Input to API
- Use of Hard-coded, Security-relevant Constants
- Unsafe Function Call from a Signal Handler
- Suspicious Comment
- Return of Stack Variable Address
- Missing Default Case in Switch Statement
- Expression Issues
- Use of Obsolete Functions
- Use of Function with Inconsistent Implementations
- Unused Variable
- Dead Code
- Resource Management Errors
- Improper Resource Shutdown or Release - (404)
- Empty Synchronized Block
- Explicit Call to Finalize()
- Reachable Assertion
- Use of Potentially Dangerous Function

Data Handling

Numeric Errors

- Use of Incorrect Byte Ordering
- Unchecked Array Indexing
- Incorrect Conversion between Numeric Types
  - Unexpected Sign Extension
  - Signed to Unsigned Conversion Error
  - Unsigned to Signed Conversion Error
  - Numeric Truncation Error
- Incorrect Calculation - (682)
  - Incorrect Calculation of Buffer Size
  - Integer Overflow or Wraparound
  - Integer Underflow (Wrap or Wraparound)
  - Off-by-one Error
  - Divide By Zero

Representation Errors

- Cleansing, Canonicalization, and Comparison Errors
- Reliance on Data/Memory Layout

Information Management Errors

- Information Leak (Information Disclosure)
  - Information Leak Through Sent Data
  - Privacy Leak through Data Queries
  - Discrepancy Information Leaks
  - Error Message Information Leak - (209)
  - Cross-boundary Cleansing: Information Leak
  - Internalized Information Leak
  - Process Environment Information Leak
  - Information Leak Through Debug Information
  - Sensitive Information Uncleared Before Release
  - Information Leak of System Data
  - Information Leak Through Caching
  - Information Leak Through Environmental Variables
  - File and Directory Information Leaks
  - Information Leak Through Query Strings in GET Request
  - Information Leak Through Indexing of Private Data
- Information Loss or Omission
- Containment Errors (Container Errors)

Improper Access of Indexable Resource ('Range Error')

Type Errors

Improper Encoding or Escaping of Output - (116)

String Errors

Data Structure Issues

Improper Handling of Syntactically Invalid Structure

Modification of Assumed-Immutable Data (MAID)

Improper Input Validation - (20)

- Pathname Traversal and Equivalence Errors
- Process Control
- Missing XML Validation
- Failure to Sanitize Data into a Different Plane ('Injection')
  - Improper Sanitization of Special Elements used in a Command ('Command Injection') - (77)
  - Failure to Preserve Web Page Structure ('Cross-site Scripting') - (79)
  - Improper Sanitization of Special Elements used in an SQL Command ('SQL Injection') - (89)
  - Failure to Sanitize Data into LDAP Queries ('LDAP Injection')
  - XML Injection (aka Blind XPath Injection)
  - Failure to Sanitize CRLF Sequences ('CRLF Injection')
  - Uncontrolled Format String
  - Failure to Sanitize Special Elements into a Different Plane
  - Argument Injection or Modification
  - Improper Control of Resource Identifiers ('Resource Injection')
  - Failure to Control Generation of Code ('Code Injection') - (94)
  - Improper Sanitization of Special Elements

- Technology-Specific Input Validation Problems
- Misinterpretation of Input
- Unchecked Input for Loop Condition
- Null Byte Interaction Error (Poison Null Byte)
- Direct Use of Unsafe JNI
- Improper Output Sanitization for Logs
- Failure to Constrain Operations within the Bounds of a Memory Buffer - (119)
- Use of Externally-Controlled Input to Select Classes or Code ('Unsafe Reflection')
- ASP.NET Misconfiguration: Not Using Input Validation Framework
- URL Redirection to Untrusted Site ('Open Redirect')
- Variable Extraction Error
- Unvalidated Function Hook Arguments
- External Control of File Name or Path - (73)
- Improper Address Validation in IOCTL with METHOD_NEITHER I/O Control Code
- Use of Path Manipulation Function without Maximum-sized Buffer

Security Features

Credentials Management

- Hard-Coded Password - (259)
- Unverified Password Change
- Missing Password Field Masking
- Weak Cryptography for Passwords
- Weak Password Requirements
- Not Using Password Aging
- Password Aging with Long Expiration
- Insufficiently Protected Credentials
- Weak Password Recovery Mechanism for Forgotten Password

Insufficient Verification of Data Authenticity

- Origin Validation Error
- Improper Verification of Cryptographic Signature
- Use of Less Trusted Source
- Acceptance of Extraneous Untrusted Data With Trusted Data
- Improperly Trusted Reverse DNS
- Insufficient Type Distinction
- Cross-Site Request Forgery (CSRF) - (352)
- Failure to Add Integrity Check Value
- Improper Validation of Integrity Check Value
- Trust of System Event Data
- Reliance on File Name or Extension of Externally-Supplied File
- Reliance on Obfuscation or Encryption of Security-Relevant Inputs without Integrity Checking

Privacy Violation

Reliance on Cookies without Validation and Integrity Checking

Client-Side Enforcement of Server-Side Security - (602)

Improperly Implemented Security Check for Standard

Improper Authentication

User Interface Security Issues

Use of Insufficiently Random Values - (330)

Logging of Excessive Data

Certificate Issues

Cryptographic Issues

- Key Management Errors
- Missing Required Cryptographic Step
- Not Using a Random IV with CBC Mode
- Failure to Encrypt Sensitive Data
  - Cleartext Storage of Sensitive Information
  - Cleartext Transmission of Sensitive Information - (319)
  - Sensitive Cookie in HTTPS Session Without 'Secure' Attribute
- Reversible One-Way Hash
- Inadequate Encryption Strength
  - Use of a Broken or Risky Cryptographic Algorithm - (327)
- Use of RSA Algorithm without OAEP

Permissions, Privileges, and Access Controls

- Access Control (Authorization) Issues - (284)
- Permission Issues
  - Incorrect Default Permissions
  - Insecure Inherited Permissions
  - Insecure Preserved Inherited Permissions
  - Incorrect Execution-Assigned Permissions
  - Improper Handling of Insufficient Permissions or Privileges
  - Improper Preservation of Permissions
  - Exposed Unsafe ActiveX Method
  - Incorrect Permission Assignment for Critical Resource - (732)
  - Permission Race Condition During Resource Copy
- Privilege / Sandbox Issues
- Improper Ownership Management
- Insecure User Management

Password in Configuration File

Insufficient Compartmentalization

Reliance on a Single Factor in a Security Decision

Insufficient Psychological Acceptability

Reliance on Security through Obscurity

Protection Mechanism Failure

Insufficient Logging

Reliance on Cookies without Validation and Integrity Checking in a Security Decision

Insufficient Encapsulation

Mobile Code Issues/Missing Custom Error Page

- Public cloneable() Method Without Final ('Object Hijack')
- Use of Inner Class Containing Sensitive Data
- Critical Public Variable Without Final Modifier
- Download of Code Without Integrity Check - (494)
- Array Declared Public, Final, and Static
- finalize() Method Declared Public

Leftover Debug Code

Use of Dynamic Class Loading

clone() Method Without super.clone()

Comparison of Classes by Name

Data Leak Between Sessions

Trust Boundary Violation

Reliance on Package-level Scope

J2EE Framework: Saving Unserializable Objects to Disk

Deserialization of Untrusted Data

Serializable Class Containing Sensitive Data

Information Leak through Class Cloning

Public Data Assigned to Private Array-Typed Field

Private Array-Typed Field Returned From A Public Method

Public Static Final Field References Mutable Object

Exposed Dangerous Method or Function

Critical Variable Declared Public

Access to Critical Private Variable via Public Method

16 July 2010

A Human Capital Crisis in Cybersecurity

Technical Proficiency Matters

A White Paper of the
CSIS Commission on Cybersecurity for the 44th Presidency

COCHAIRS
Representative James R. Langevin
Representative Michael T. McCaul
Scott Charney
Lt. General Harry Raduege, USAF (ret.)

based on a body of knowledge that represents the complete set of concepts, terms and activities that make up a professional domain. And absent such a body of knowledge there is little basis for supporting a certification program. Indeed it would be dangerous and misleading.

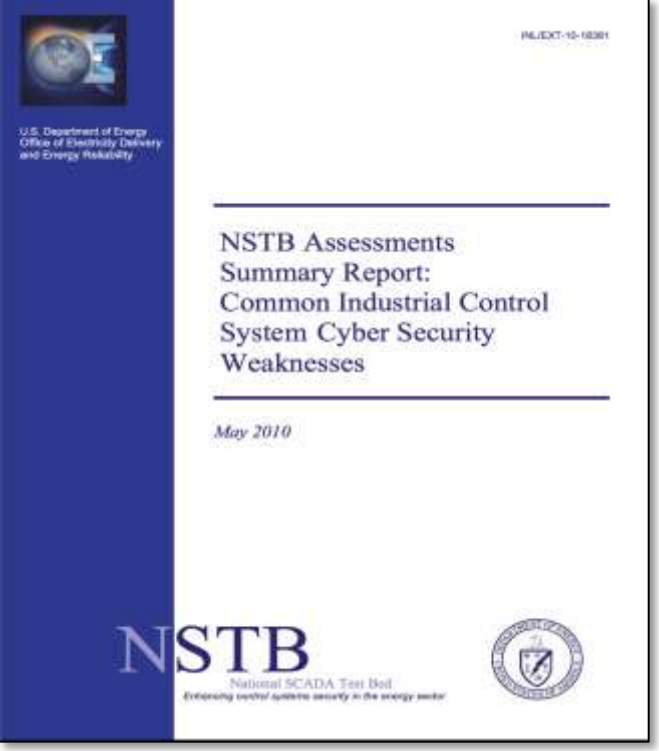
A complete body of knowledge covering the entire field of software engineering may be years away. However, the body of knowledge needed by professionals to create software free of common and critical security flaws has been developed, vetted widely and kept up to date. That is the foundation for a certification 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.

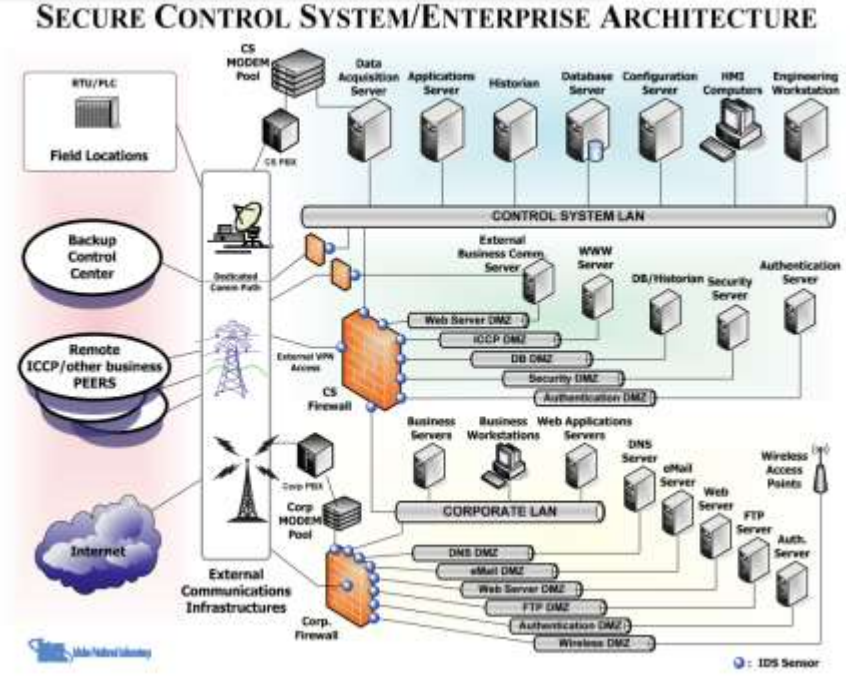
Any programmer who writes code without being aware of those problems and is not capable of writing code free of those errors is a threat to his or her employers and to others who use computers connected to systems running his or her software.

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.

Idaho National Labs SCADA Report





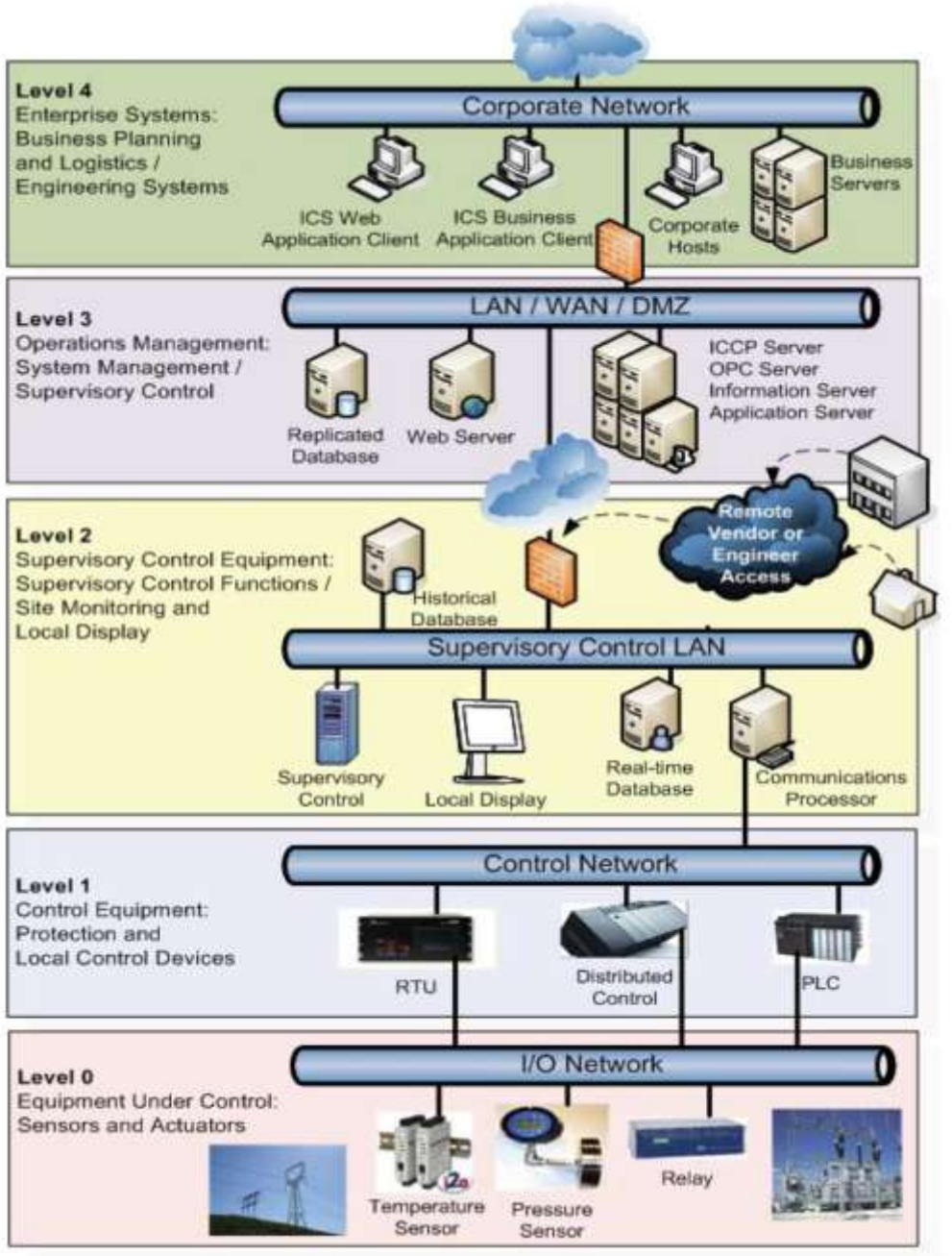


Table 27. Most common programming errors found in ICS code.

Weakness Classification	Vulnerability Type
CWE-19: Data Handling	CWE-228: Improper Handling of Syntactically Invalid Structure
	CWE-229: Improper Handling of Values
	CWE-230: Improper Handling of Missing Values
	CWE-20: Improper Input Validation
	CWE-116: Improper Encoding or Escaping of Output
	CWE-195: Signed to Unsigned Conversion Error
	CWE-198: Use of Incorrect Byte Ordering
CWE-119: Failure to Constrain Operations within the Bounds of a Memory Buffer	CWE-120: Buffer Copy without Checking Size of Input ("Classic Buffer Overflow")
	CWE-121: Stack-based Buffer Overflow
	CWE-122: Heap-based Buffer Overflow
	CWE-125: Out-of-bounds Read
	CWE-129: Improper Validation of Array Index
	CWE-131: Incorrect Calculation of Buffer Size
	CWE-170: Improper Null Termination
	CWE-190: Integer Overflow or Wraparound
	CWE-680: Integer Overflow to Buffer Overflow
CWE-398: Indicator of Poor Code Quality	CWE-454: External Initialization of Trusted Variables or Data Stores
	CWE-456: Missing Initialization
	CWE-457: Use of Uninitialized Variable
	CWE-476: NULL Pointer Dereference
	CWE-400: Uncontrolled Resource Consumption ("Resource Exhaustion")
	CWE-252: Unchecked Return Value
	CWE-690: Unchecked Return Value to NULL Pointer Dereference
	CWE-772: Missing Release of Resource after Effective Lifetime
CWE-442: Web Problems	CWE-22: Improper Limitation of a Pathname to a Restricted Directory ("Path Traversal")
	CWE-79: Failure to Preserve Web Page Structure ("Cross-site Scripting")
	CWE-89: Failure to Preserve SQL Query Structure ("SQL Injection")
CWE-703: Failure to Handle Exceptional Conditions	CWE-431: Missing Handler
	CWE-248: Uncaught Exception
	CWE-755: Improper Handling of Exceptional Conditions
	CWE-390: Detection of Error Condition Without Action

Linkage with Fundamental Changes in Enterprise Security Initiatives

Twenty Critical Controls for Effective Cyber Def Guidelines

What the 20 CSC Critics say...

20 Critical Security Controls - Version 2.0

- 20 Critical Security Controls - Introduction (Version 2.0)
- Critical Control 1: Inventory of Authorized and Unauthorized
- Critical Control 2: Inventory of Authorized and Unauthorized
- Critical Control 3: Secure Configurations for Hardware and So Servers
- Critical Control 4: Secure Configurations for Network Devices
- Critical Control 5: Boundary Defense
- Critical Control 6: Maintenance, Monitoring, and Analysis of
- Critical Control 7: Application Software Security
- Critical Control 8: Controlled Use of Administrative Privilege
- Critical Control 9: Controlled Access Based on Need to Know
- Critical Contro
- Critical Contro
- Critical Contro
- Critical Contro
- Critical Contro
- Critical Contro
- Critical Contro
- Critical Contro
- Critical Contro
- Critical Contro
- Critical Contro

CAG: Critical Control 7: Application Software Security

<< previous control | Consensus Audit Guidelines | next control >>

How do attackers exploit the lack of this control?

Attacks against vulnerabilities in web-based and other application software have been a top priority for criminal organizations in recent years. Application software that does not properly check the size of user input, fails to sanitize user input by filtering out unneeded but potentially malicious character sequences, or does not initialize and clear variables properly could be vulnerable to remote compromise. Attackers can inject specific exploits, including buffer overflows, SQL injection attacks, and cross-site scripting code to gain control over vulnerable machines. In one attack in 2008, more than 1 million web servers were exploited and turned into infection engines for visitors to those sites using SQL injection. During that attack, trusted websites from state governments and other organizations compromised by attackers were used to infect hundreds of thousands of

CWE and CAPEC included in Control 7 of the "Twenty Critical Controls for Effective Cyber Defense: Consensus Audit Guidelines"

Procedures and tools for implementing t

Source code testing tools, web application security scanning tools, and object code testing tools have proven useful in securing application software, along with manual application security penetration testing by testers who have extensive programming knowledge as well as application penetration testing expertise. The Common Weakness Enumeration (CWE) initiative is utilized by many such tools to identify the weaknesses that they find. Organizations can also use CWE to determine which types of weaknesses they are most interested in addressing and removing. A broad community effort to identify the "Top 25 Most Dangerous Programming Errors" is also available as a minimum set of important issues to investigate and address during the application development process. When evaluating the effectiveness of testing for these weaknesses, the Common Attack Pattern Enumeration and Classification (CAPEC) can be used to organize and record the breadth of the testing for the CWEs as well as a way for testers to think like attackers in their development of test cases.

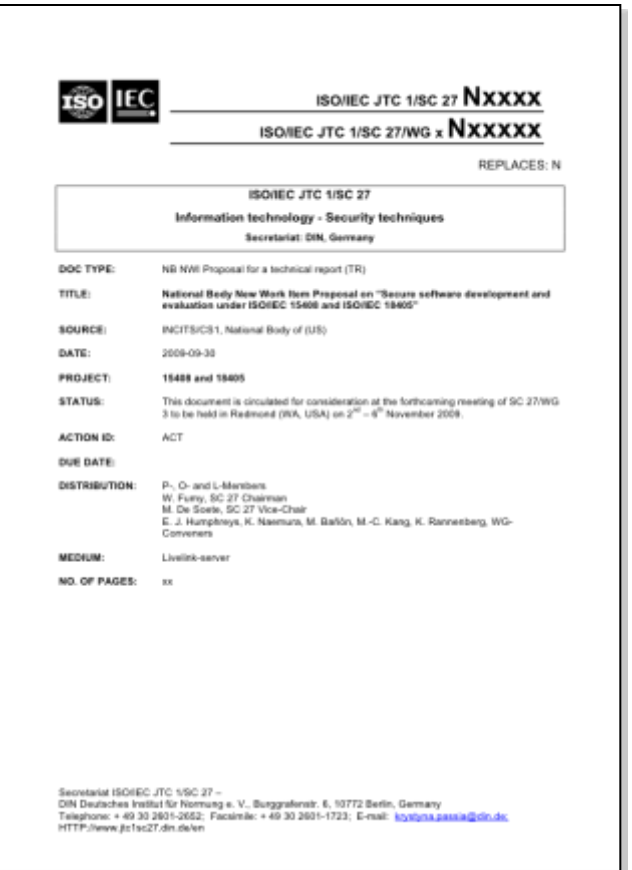
SC27 WG3

Common Criteria v4 CCDB

- TOE to leverage CAPEC & CWE
- Also investigating how to leverage ISO/IEC 15026

NIAP Evaluation Scheme

- Above plus
- Also investigating how to leverage SCAP



ISO/IEC JTC 1/SC 27 **Nxxxx**

ISO/IEC JTC 1/SC 27/WG x **Nxxxxx**

REPLACES: N

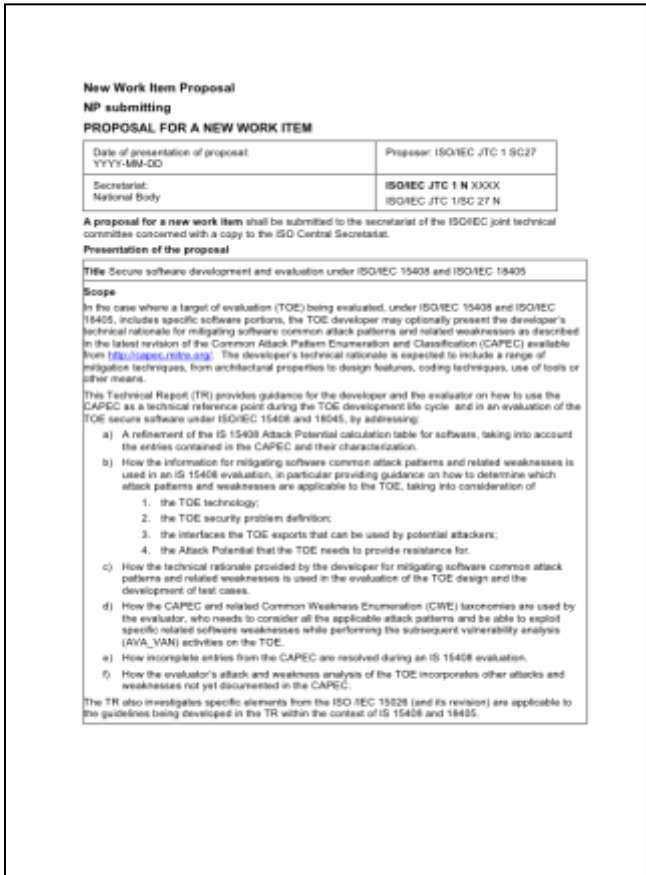
ISO/IEC JTC 1/SC 27

Information technology - Security techniques

Secretariat: DIN, Germany

DOC TYPE:	NB NWI Proposal for a technical report (TR)
TITLE:	**National Body New Work Item Proposal on "Secure software development and evaluation under ISO/IEC 15408 and ISO/IEC 18405"**
SOURCE:	INCITS/CS1, National Body of (US)
DATE:	2009-09-30
PROJECT:	**15408 and 18405**
STATUS:	This document is circulated for consideration at the forthcoming meeting of SC 27/WG 3 to be held in Redmond (WA, USA) on 2nd – 6th November 2009.
ACTION ID:	ACT
DUE DATE:	
DISTRIBUTION:	P-, O- and L-Members W. Fumy, SC 27 Chairman M. De Soete, SC 27 Vice-Chair E. J. Humphreys, K. Naemura, M. Bañón, M.-C. Kang, K. Rannenberg, WG-Convenors
MEDIUM:	Livelink-server
NO. OF PAGES:	xx

Secretariat ISO/IEC JTC 1/SC 27 –
DIN Deutsches Institut für Normung e. V., Burggrafenstr. 6, 10772 Berlin, Germany
Telephone: + 49 30 2601-2652; Facsimile: + 49 30 2601-1723; E-mail: krystyna.passia@din.de
HTTP://www.jtc1sc27.din.de/en



New Work Item Proposal

NP submitting

PROPOSAL FOR A NEW WORK ITEM

Date of presentation of proposal: YYYY-MM-DD	Proposer: ISO/IEC JTC 1 SC27
Secretariat: National Body	**ISO/IEC JTC 1 N** XXXX ISO/IEC JTC 1/SC 27 N

A proposal for a new work item shall be submitted to the secretariat of the ISO/IEC joint technical committee concerned with a copy to the ISO Central Secretariat.

Presentation of the proposal

Title Secure software development and evaluation under ISO/IEC 15408 and ISO/IEC 18405

Scope

In the case where a target of evaluation (TOE) being evaluated, under ISO/IEC 15408 and ISO/IEC 18405, includes specific software portions, the TOE developer may optionally present the developer's technical rationale for mitigating software common attack patterns and related weaknesses as described in the latest revision of the Common Attack Pattern Enumeration and Classification (CAPEC) available from http://capec.mitre.org/. The developer's technical rationale is expected to include a range of mitigation techniques, from architectural properties to design features, coding techniques, use of tools or other means.

This Technical Report (TR) provides guidance for the developer and the evaluator on how to use the CAPEC as a technical reference point during the TOE development life cycle and in an evaluation of the TOE secure software under ISO/IEC 15408 and 18045, by addressing:

a) A refinement of the IS 15408 Attack Potential calculation table for software, taking into account the entries contained in the CAPEC and their characterization.

b) How the information for mitigating software common attack patterns and related weaknesses is used in an IS 15408 evaluation, in particular providing guidance on how to determine which attack patterns and weaknesses are applicable to the TOE, taking into consideration of

1. the TOE technology;
2. the TOE security problem definition;
3. the interfaces the TOE exports that can be used by potential attackers;
4. the Attack Potential that the TOE needs to provide resistance for.

c) How the technical rationale provided by the developer for mitigating software common attack patterns and related weaknesses is used in the evaluation of the TOE design and the development of test cases.

d) How the CAPEC and related Common Weakness Enumeration (CWE) taxonomies are used by the evaluator, who needs to consider all the applicable attack patterns and be able to exploit specific related software weaknesses while performing the subsequent vulnerability analysis (AVA_VAN) activities on the TOE.

e) How incomplete entries from the CAPEC are resolved during an IS 15408 evaluation.

f) How the evaluator's attack and weakness analysis of the TOE incorporates other attacks and weaknesses not yet documented in the CAPEC.

The TR also investigates specific elements from the ISO /IEC 15026 (and its revision) are applicable to the guidelines being developed in the TR within the context of IS 15408 and 18405.


Questions?
ramartin@mitre.org