Measuring Software Security

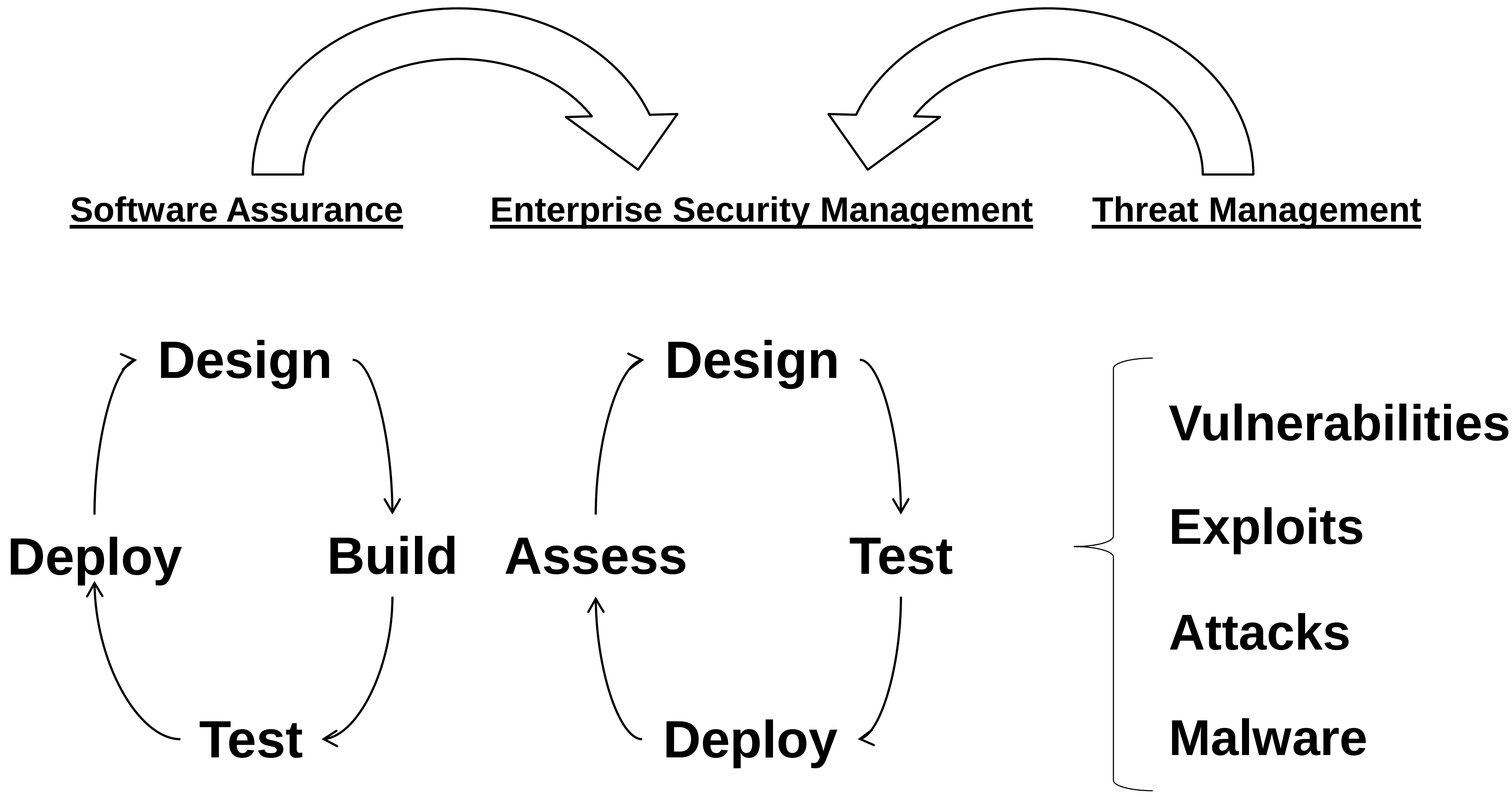
Automation and Standards to Assure that we "Build Security In"

Oct 31, 2011

Sean Barnum

Making Security Measurable (MSM)

“You Are Here”



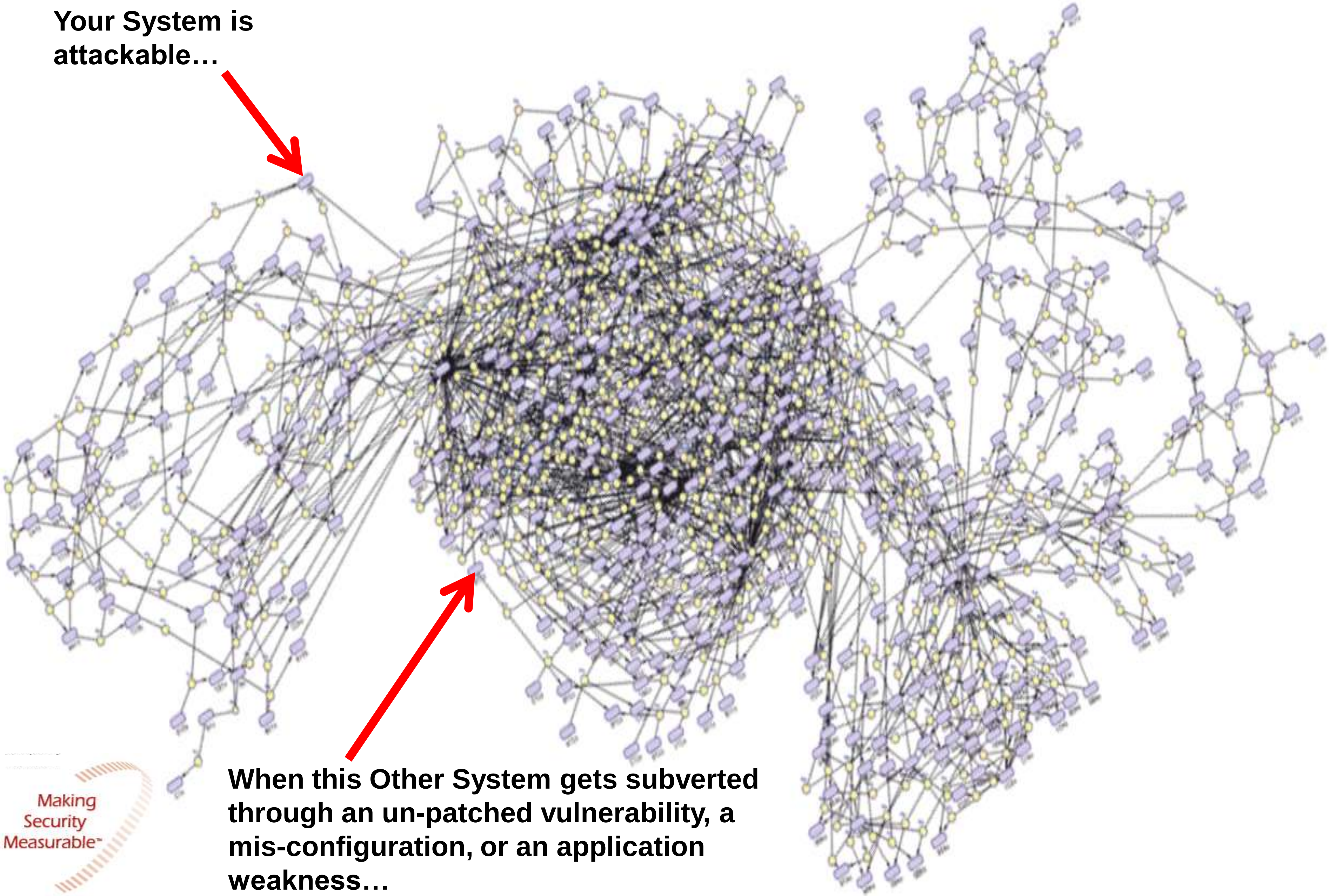
CWE, CAPEC, CWSS, CWRAF

CPE, CCE, OVAL, OCIL, XCCDF, AssetId, ARF

CVE, CWE, CAPEC, MAEC, CybOX, IODEF, RID, RID-T, CYBEX

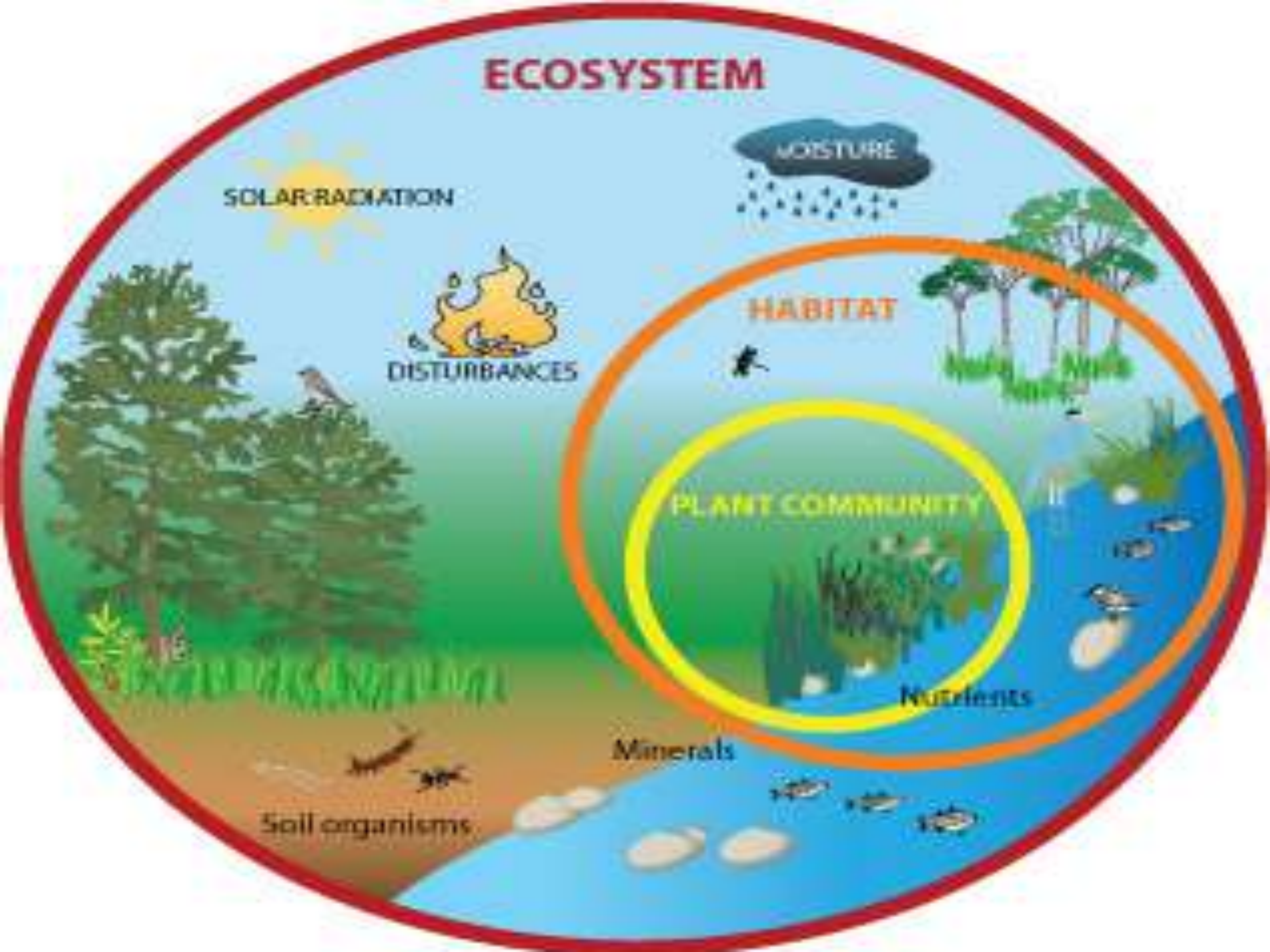
Today Everything's Connected – Like an Ecosystem

Your System is attackable…

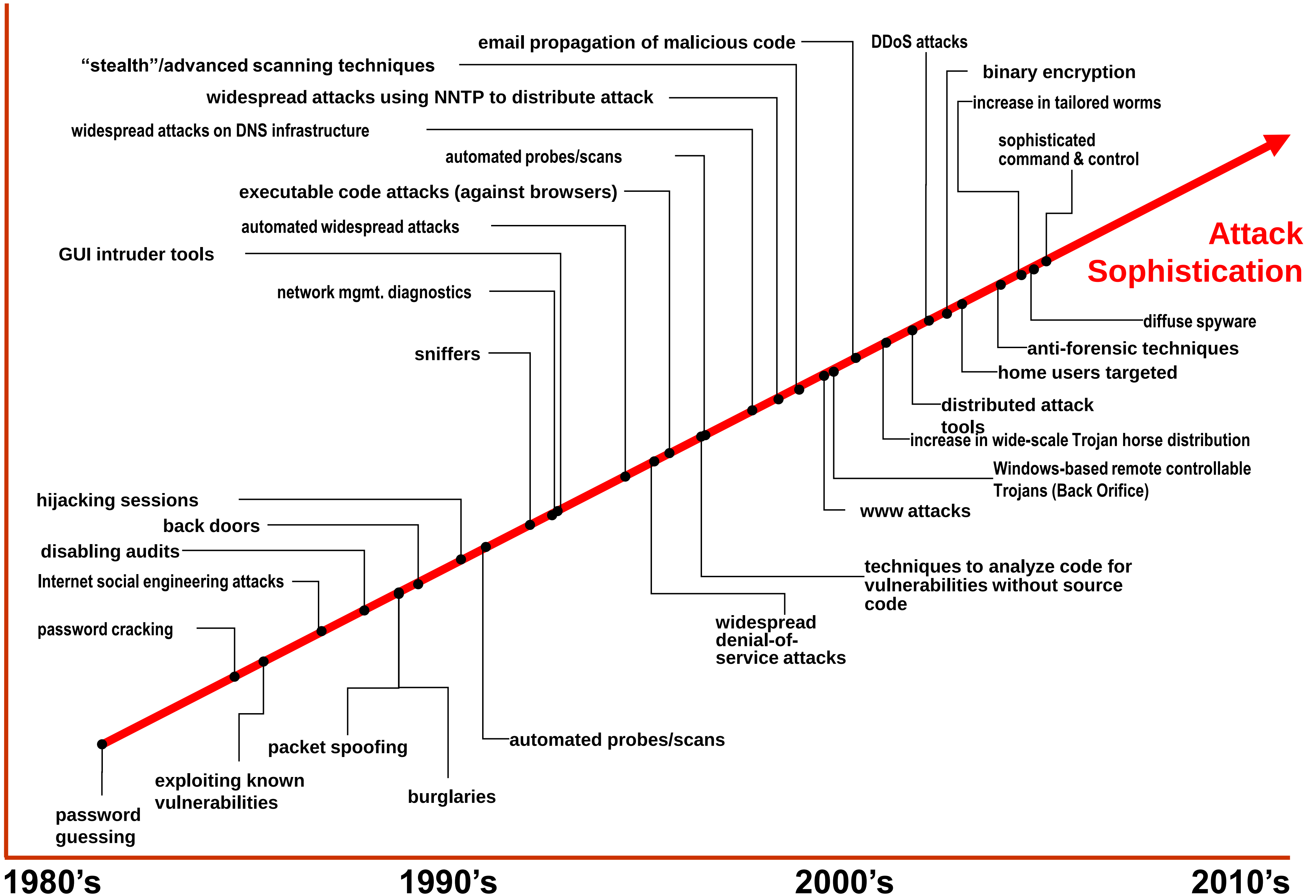


When this Other System gets subverted through an un-patched vulnerability, a mis-configuration, or an application weakness…

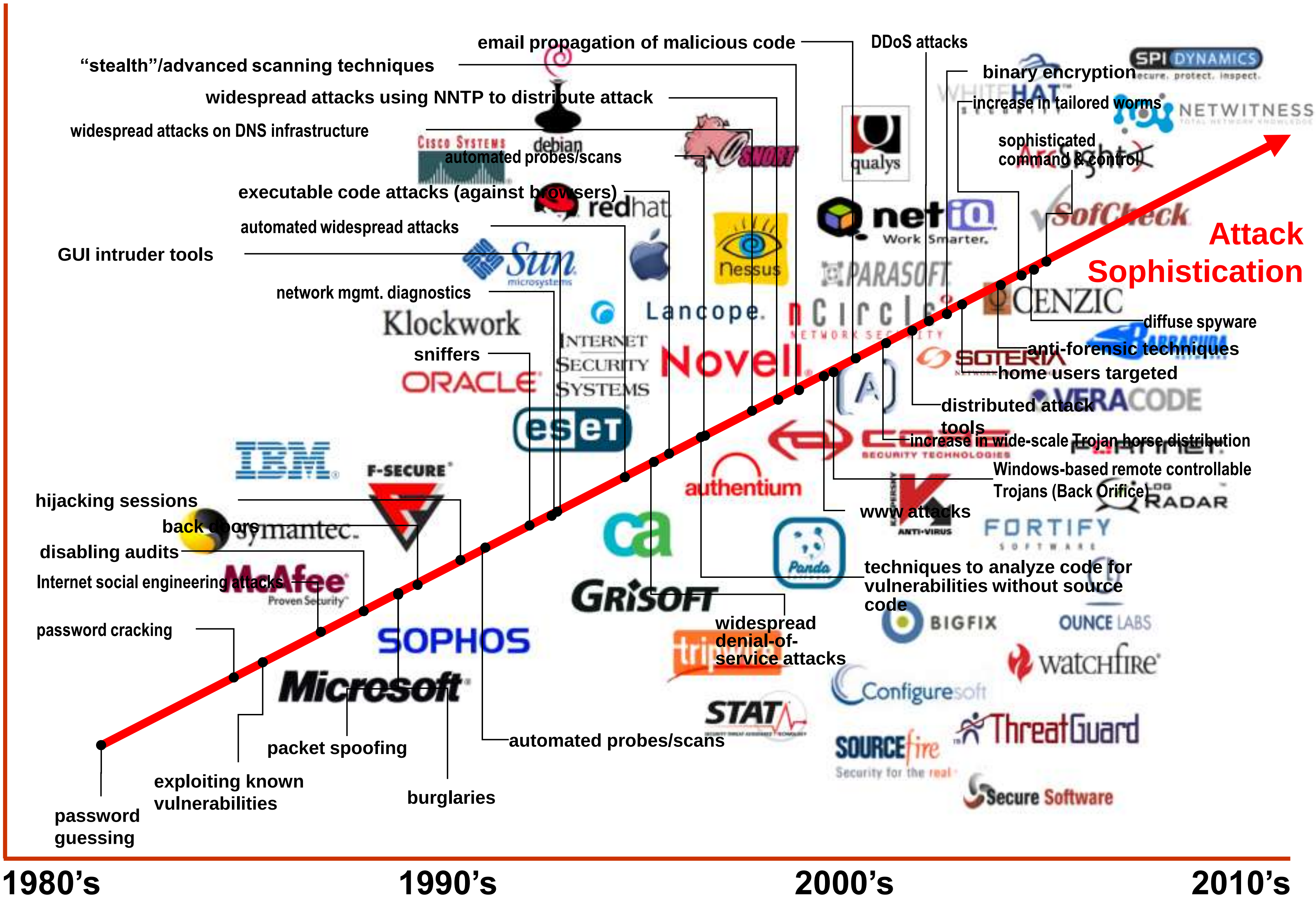
Making Security Measurable™


ECOSYSTEM
MOISTURE
SOLAR RADIATION
HABITAT
DISTURBANCES
PLANT COMMUNITY
Nutrients
Minerals
Soil organisms

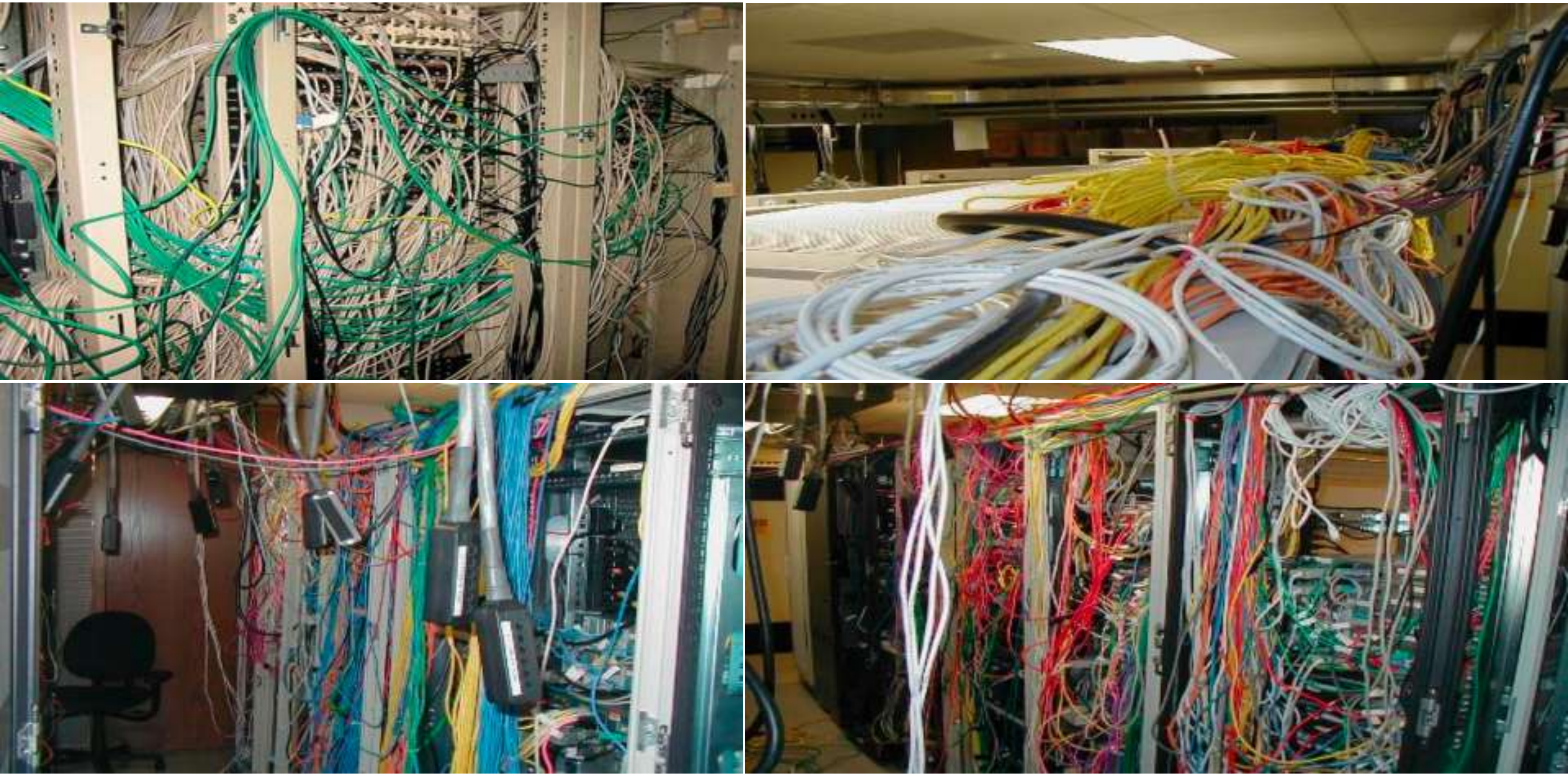
Our Awareness of Cyber Threats Emerged Over Time



Applying Solutions Also Evolved Over Time



Like Our Security Solutions - Networks Evolved



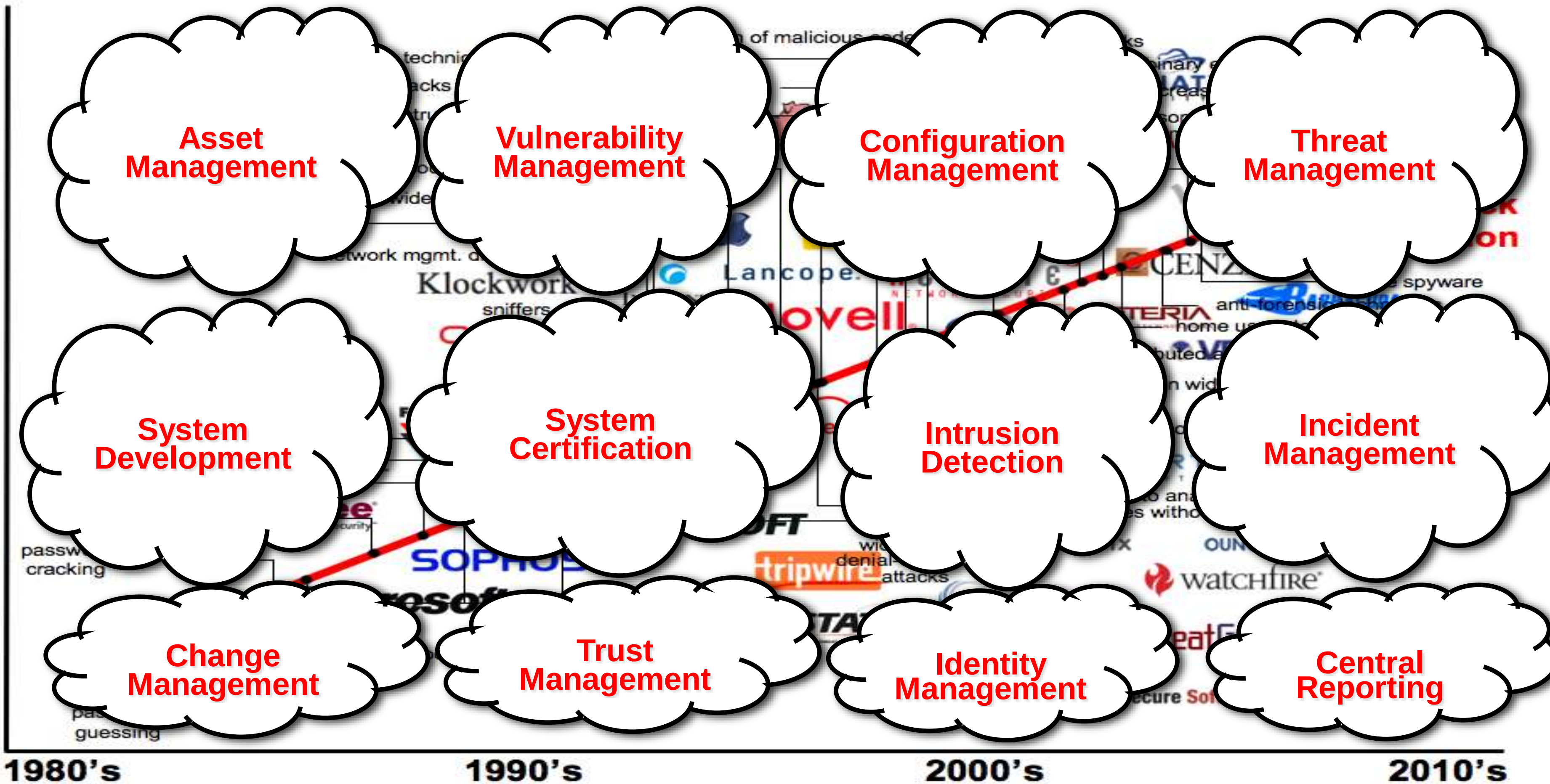
Each new solution had to integrate with the existing solutions

-->> every enterprise ends up learning as they go and has a "unique" tapestry of solutions with "local practices"

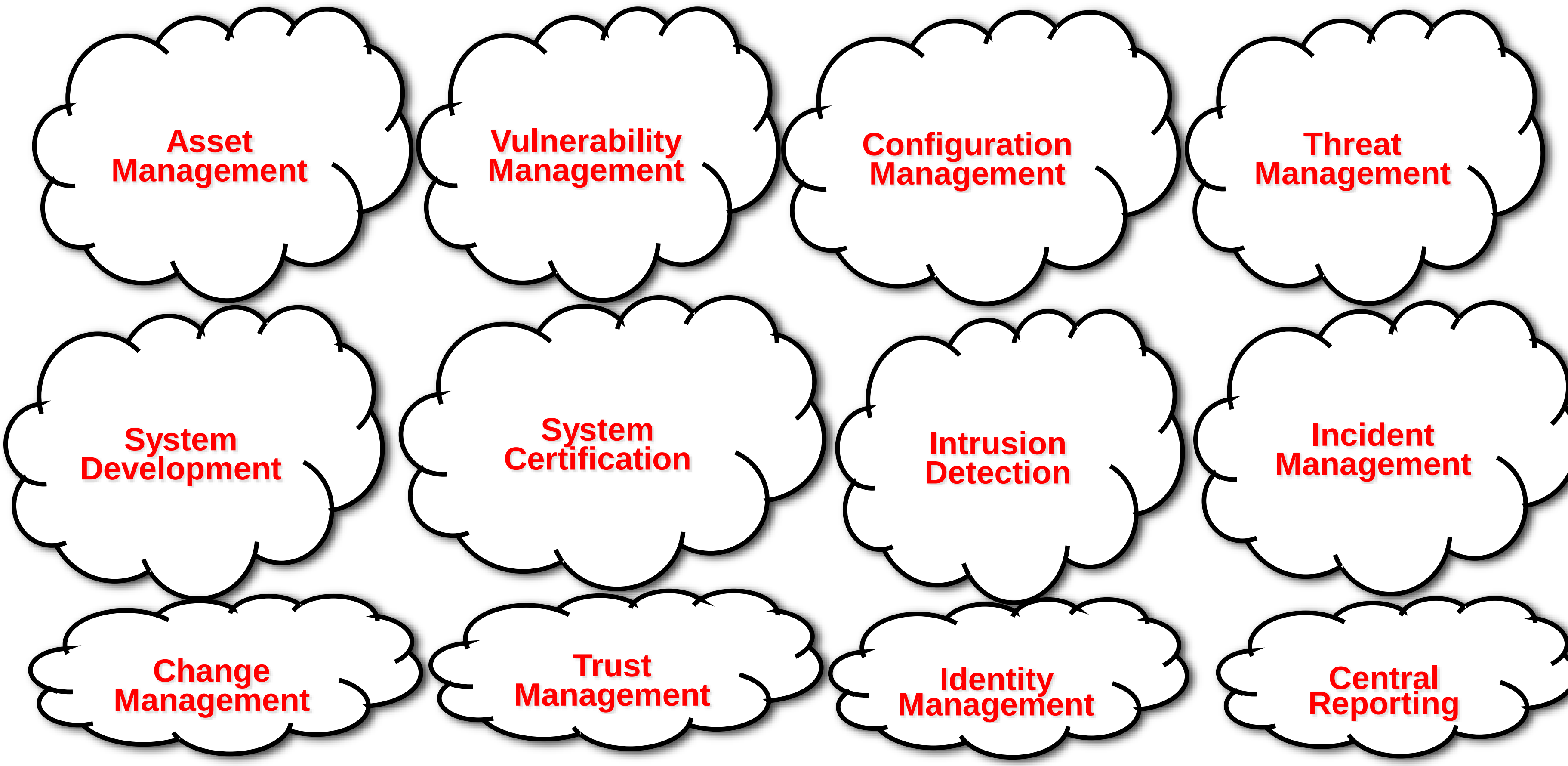
But A More Supportable Solution Is Possible with Standardized Approaches and the application of Architecting Principles

Making Security Measurable™

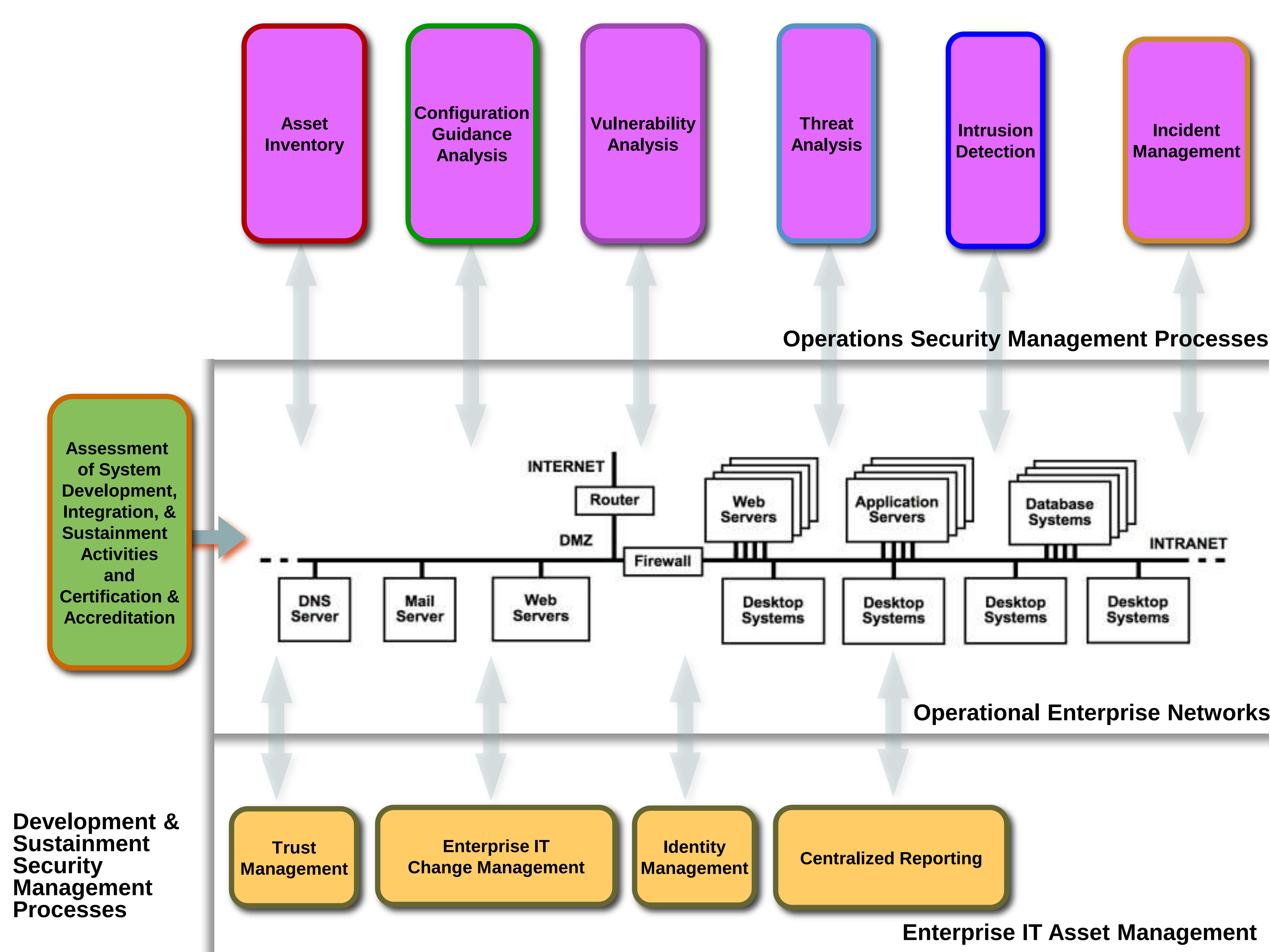
Architecting Security with Information Standards for Communities of Interest

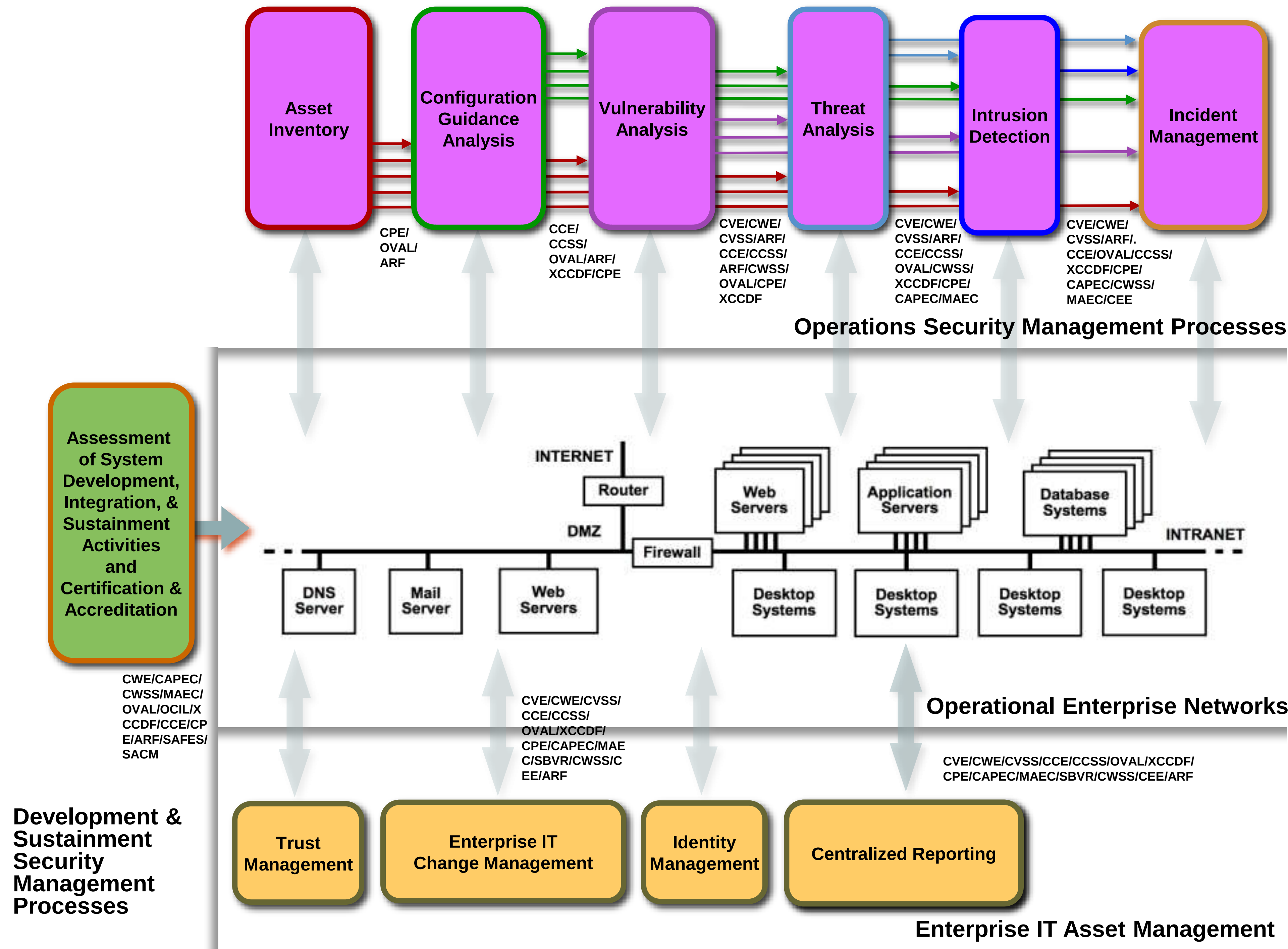


Making Security Measurable



Making Security Measurable


Asset Inventory
Configuration Guidance Analysis
Vulnerability Analysis
Threat Analysis
Intrusion Detection
Incident Management
Operations Security Management Processes
Assessment of System Development, Integration, & Sustainment Activities and Certification & Accreditation
INTERNET
Router
DMZ
Firewall
Web Servers
Application Servers
Database Systems
INTRANET
DNS Server
Mail Server
Web Servers
Desktop Systems
Desktop Systems
Desktop Systems
Desktop Systems
Operational Enterprise Networks
Development & Sustainment Security Management Processes
Trust Management
Enterprise IT Change Management
Identity Management
Centralized Reporting
Enterprise IT Asset Management


Asset Inventory
Configuration Guidance Analysis
Vulnerability Analysis
Threat Analysis
Intrusion Detection
Incident Management
CPE/ OVAL/ ARF
CCE/ CCSS/ OVAL/ARF/ XCCDF/CPE
CVE/CWE/ CVSS/ARF/ CCE/CCSS/ ARF/CWSS/ OVAL/CPE/ XCCDF
CVE/CWE/ CVSS/ARF/ CCE/CCSS/ OVAL/CWSS/ XCCDF/CPE/ CAPEC/MAEC
CVE/CWE/ CVSS/ARF/. CCE/OVAL/CCSS/ XCCDF/CPE/ CAPEC/CWSS/ MAEC/CEE
Operations Security Management Processes
Assessment of System Development, Integration, & Sustainment Activities and Certification & Accreditation
INTERNET
Router
DMZ
Firewall
INTRANET
Web Servers
Application Servers
Database Systems
DNS Server
Mail Server
Web Servers
Desktop Systems
Desktop Systems
Desktop Systems
Desktop Systems
CWE/CAPEC/ CWSS/MAEC/ OVAL/OCIL/X CCDF/CCE/CP E/ARF/SAFES/ SACM
CVE/CWE/CVSS/ CCE/CCSS/ OVAL/XCCDF/ CPE/CAPEC/MAE C/SBVR/CWSS/C EE/ARF
Operational Enterprise Networks
CVE/CWE/CVSS/CCE/CCSS/OVAL/XCCDF/ CPE/CAPEC/MAEC/SBVR/CWSS/CEE/ARF
Development & Sustainment Security Management Processes
Trust Management
Enterprise IT Change Management
Identity Management
Centralized Reporting
Enterprise IT Asset Management

Cyber Ecosystem Standardization Efforts

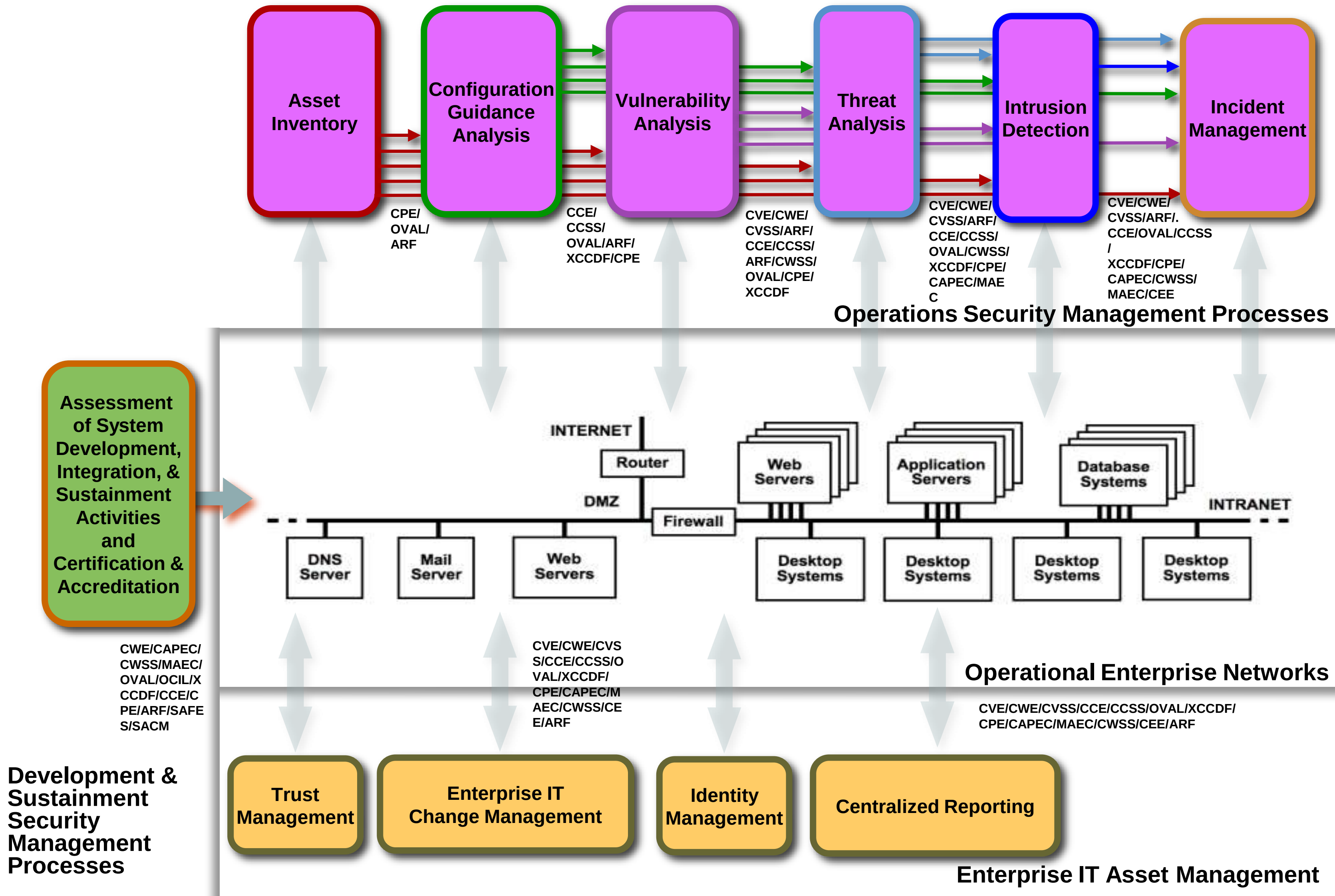
What IT systems do I have in my enterprise?	• CPE (Platforms)
What known vulnerabilities do I need to worry about?	• CVE (Vulnerabilities)
What vulnerabilities do I need to worry about right now?	• CVSS (Scoring System)
How can I configure my systems more securely?	• CCE (Configurations)
How do I define a policy of secure configurations?	• XCCDF (Configuration Checklists)
How can I be sure my systems conform to policy?	• OVAL (Assessment Language)
How can I be sure the operation of my systems conforms to policy?	• OCIL (Interactive Language)
What weaknesses in my software could be exploited?	• CWE (Weaknesses)
What attacks can exploit which weaknesses?	• CAPEC (Attack Patterns)
How can we recognize malware & share that info?	• MAEC (Malware Attributes)
What observable behavior might put my enterprise at risk?	• CybOX (Cyber Observables)
What events should be logged, and how?	• CEE (Events)
How can I aggregate assessment results?	• ARF (Assessment Results)

Standardization Efforts leveraged by the Security Content Automation Protocol (SCAP)

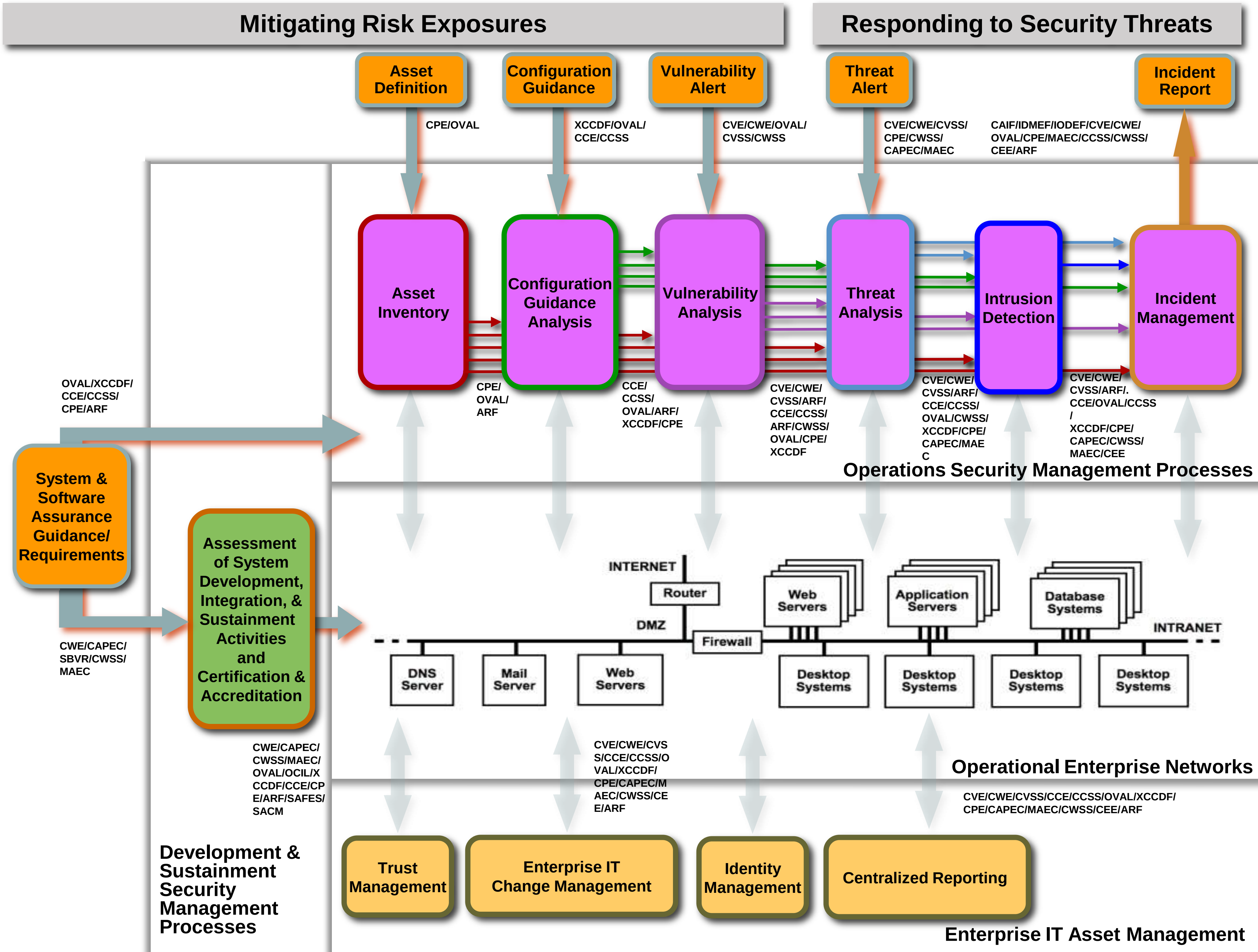
What IT systems do I have in my enterprise?	• CPE (Platforms)
What known vulnerabilities do I need to worry about?	• CVE (Vulnerabilities)
What vulnerabilities do I need to worry about right now?	• CVSS (Scoring System)
How can I configure my systems more securely?	• CCE (Configurations)
How do I define a policy of secure configurations?	• XCCDF (Configuration Checklists)
How can I be sure my systems conform to policy?	• OVAL (Assessment Language)
How can I be sure the operation of my systems conforms to policy?	• OCIL (Interactive Language)
What weaknesses in my software could be exploited?	• CWE (Weaknesses)
What attacks can exploit which weaknesses?	• CAPEC (Attack Patterns)
How can we recognize malware & share that info?	• MAEC (Malware Attributes)
What observable behavior might put my enterprise at risk?	• CybOX (Cyber Observables)
What events should be logged, and how?	• CEE (Events)
How can I aggregate assessment results?	• ARF (Assessment Results)

Standardization Efforts focused on mitigating risks and enabling faster incident response

What IT systems do I have in my enterprise?	• CPE (Platforms)
What known vulnerabilities do I need to worry about?	• CVE (Vulnerabilities)
What vulnerabilities do I need to worry about right now?	• CVSS (Scoring System)
How can I configure my systems more securely?	• CCE (Configurations)
How do I define a policy of secure configurations?	• XCCDF (Configuration Checklists)
How can I be sure my systems conform to policy?	• OVAL (Assessment Language)
How can I be sure the operation of my systems conforms to policy?	• OCIL (Interactive Language)
What weaknesses in my software could be exploited?	• CWE (Weaknesses)
What attacks can exploit which weaknesses?	• CAPEC (Attack Patterns)
How can we recognize malware & share that info?	• MAEC (Malware Attributes)
What observable behavior might put my enterprise at risk?	• CybOX (Cyber Observables)
What events should be logged, and how?	• CEE (Events)
How can I aggregate assessment results?	• ARF (Assessment Results)

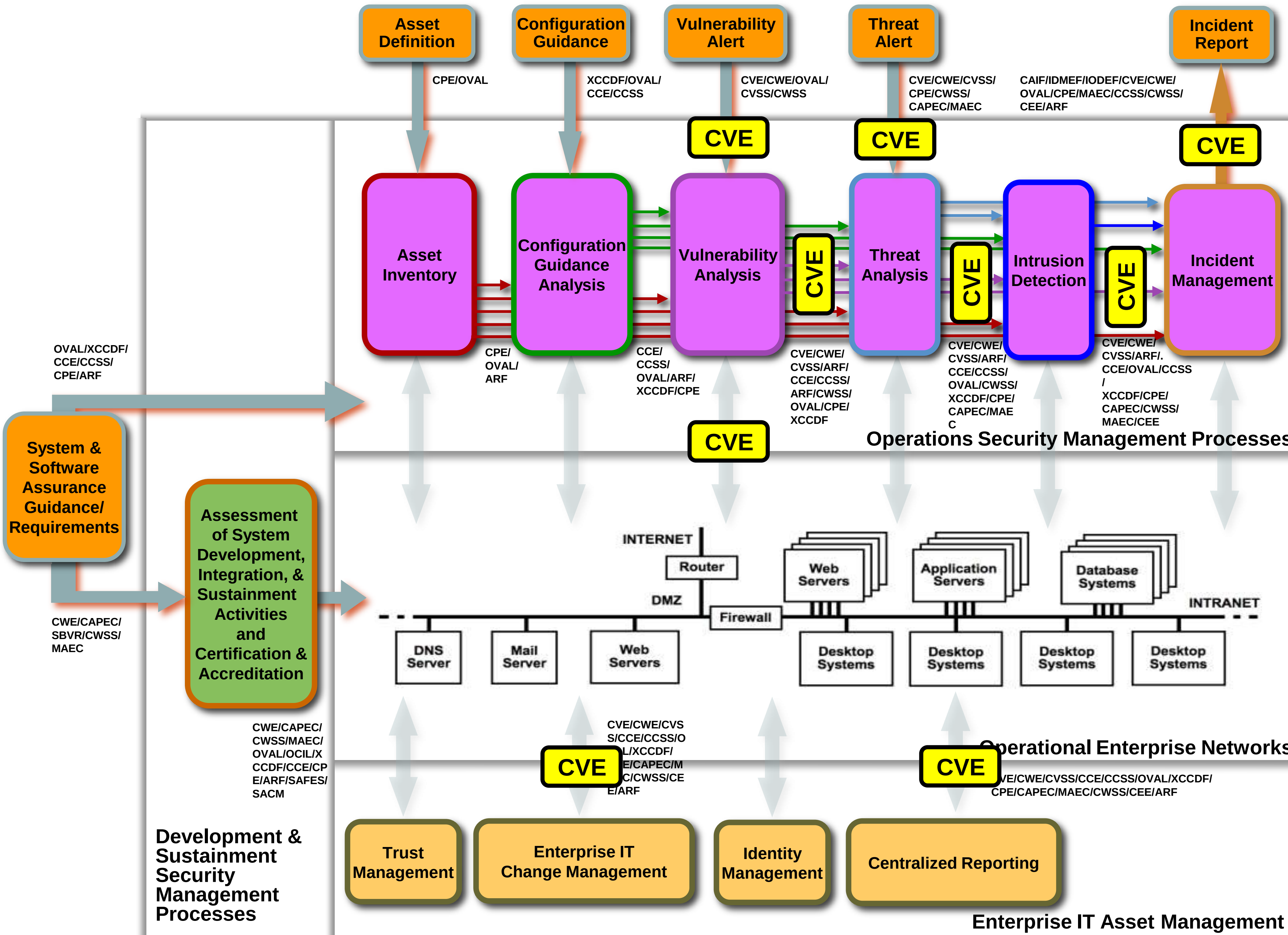


Asset Inventory
Configuration Guidance Analysis
Vulnerability Analysis
Threat Analysis
Intrusion Detection
Incident Management
CPE/OVAL/ARF
CCE/CCSS/OVAL/ARF/XCCDF/CPE
CVE/CWE/CVSS/ARF/CCE/CCSS/ARF/CWSS/OVAL/CPE/XCCDF
CVE/CWE/CVSS/ARF/CCE/CCSS/OVAL/CWSS/XCCDF/CPE/CAPEC/MAEC
CVE/CWE/CVSS/ARF/.CCE/OVAL/CCSS/XCCDF/CPE/CAPEC/CWSS/MAEC/CEE
Operations Security Management Processes
Assessment of System Development, Integration, & Sustainment Activities and Certification & Accreditation
INTERNET
Router
DMZ
Firewall
Web Servers
Application Servers
Database Systems
INTRANET
DNS Server
Mail Server
Web Servers
Desktop Systems
Desktop Systems
Desktop Systems
Desktop Systems
CWE/CAPEC/CWSS/MAEC/OVAL/OCIL/XCCDF/CCE/CPE/ARF/SAFES/SACM
CVE/CWE/CVSS/CCE/CCSS/OVAL/XCCDF/CPE/CAPEC/MAEC/CWSS/CEE/ARF
Operational Enterprise Networks
CVE/CWE/CVSS/CCE/CCSS/OVAL/XCCDF/CPE/CAPEC/MAEC/CWSS/CEE/ARF
Development & Sustainment Security Management Processes
Trust Management
Enterprise IT Change Management
Identity Management
Centralized Reporting
Enterprise IT Asset Management

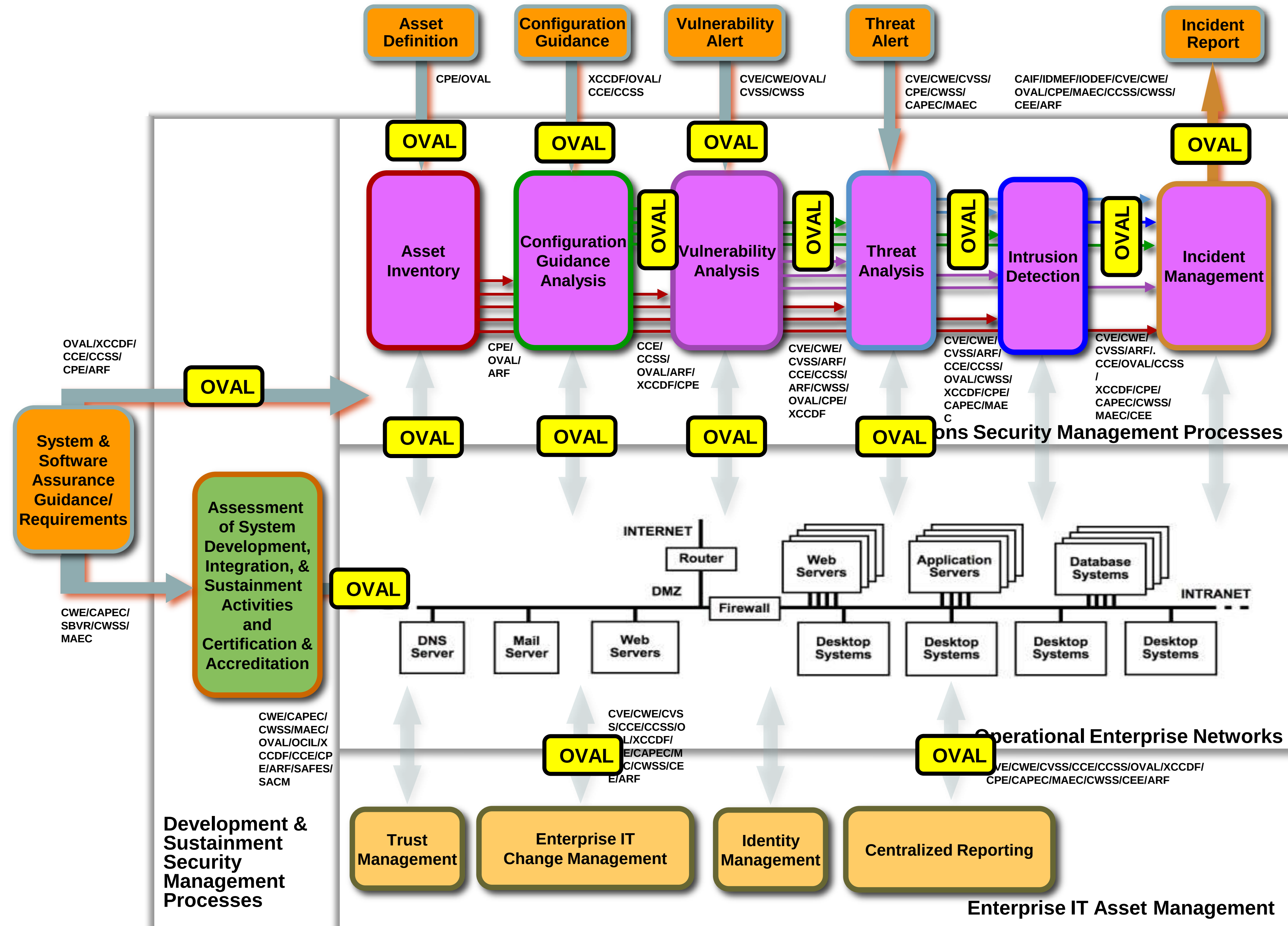


Mitigating Risk Exposures
Responding to Security Threats
Asset Definition
Configuration Guidance
Vulnerability Alert
Threat Alert
Incident Report
CPE/OVAL
XCCDF/OVAL/ CCE/CCSS
CVE/CWE/OVAL/ CVSS/CWSS
CVE/CWE/CVSS/ CPE/CWSS/ CAPEC/MAEC
CAIF/IDMEF/IODEF/CVE/CWE/ OVAL/CPE/MAEC/CCSS/CWSS/ CEE/ARF
Asset Inventory
Configuration Guidance Analysis
Vulnerability Analysis
Threat Analysis
Intrusion Detection
Incident Management
OVAL/XCCDF/ CCE/CCSS/ CPE/ARF
CPE/ OVAL/ ARF
CCE/ CCSS/ OVAL/ARF/ XCCDF/CPE
CVE/CWE/ CVSS/ARF/ CCE/CCSS/ ARF/CWSS/ OVAL/CPE/ XCCDF
CVE/CWE/ CVSS/ARF/ CCE/CCSS/ OVAL/CWSS/ XCCDF/CPE/ CAPEC/MAEC
CVE/CWE/ CVSS/ARF/. CCE/OVAL/CCSS / XCCDF/CPE/ CAPEC/CWSS/ MAEC/CEE
Operations Security Management Processes
System & Software Assurance Guidance/ Requirements
Assessment of System Development, Integration, & Sustainment Activities and Certification & Accreditation
CWE/CAPEC/ SBVR/CWSS/ MAEC
INTERNET
Router
DMZ
Firewall
Web Servers
Application Servers
Database Systems
INTRANET
DNS Server
Mail Server
Web Servers
Desktop Systems
Desktop Systems
Desktop Systems
Desktop Systems
CWE/CAPEC/ CWSS/MAEC/ OVAL/OCIL/X CCDF/CCE/CP E/ARF/SAFES/ SACM
CVE/CWE/CVS S/CCE/CCSS/O VAL/XCCDF/ CPE/CAPEC/M AEC/CWSS/CE E/ARF
Operational Enterprise Networks
CVE/CWE/CVSS/CCE/CCSS/OVAL/XCCDF/ CPE/CAPEC/MAEC/CWSS/CEE/ARF
Development & Sustainment Security Management Processes
Trust Management
Enterprise IT Change Management
Identity Management
Centralized Reporting
Enterprise IT Asset Management

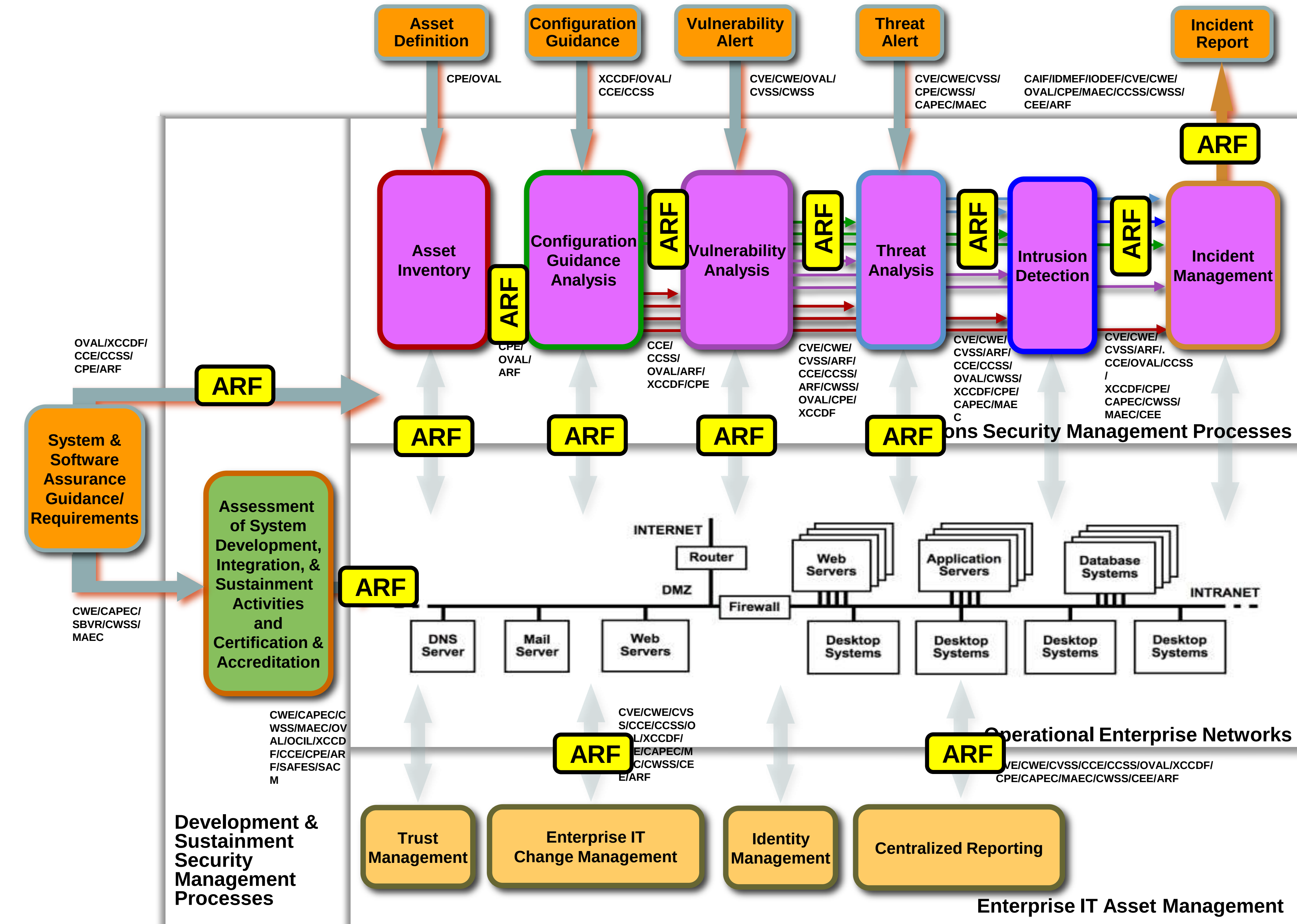
Knowledge Repositories



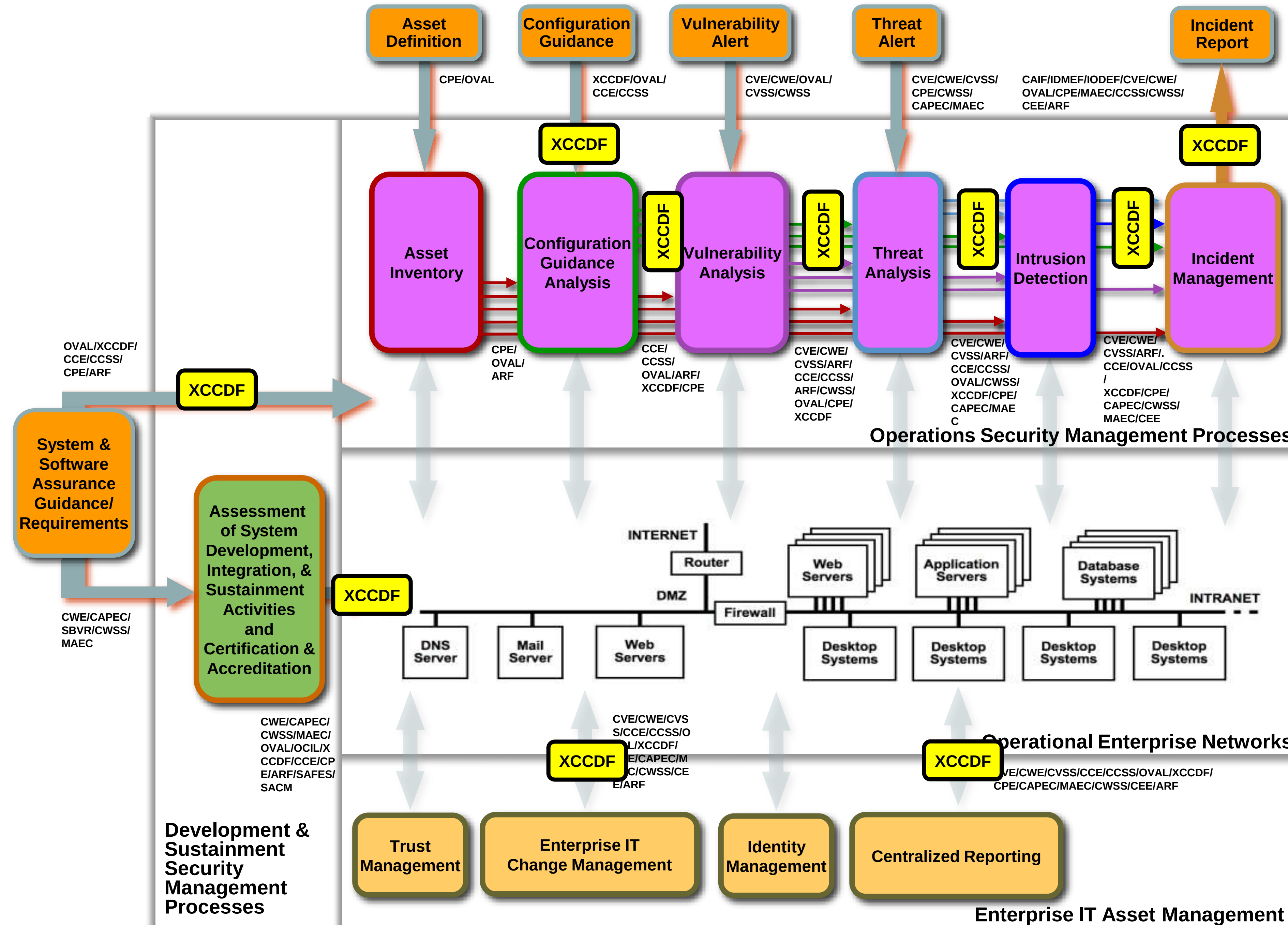
Knowledge Repositories



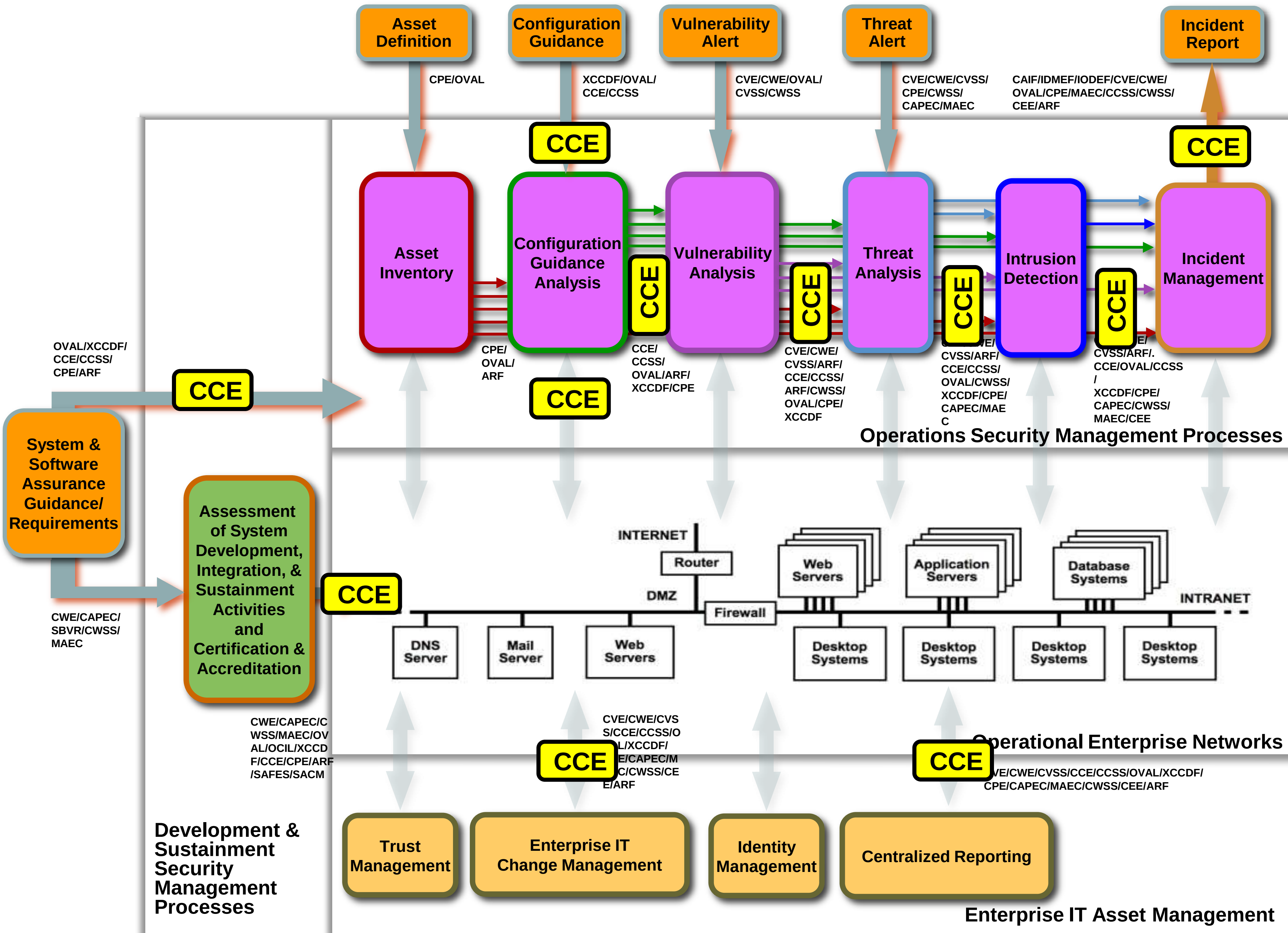
Knowledge Repositories



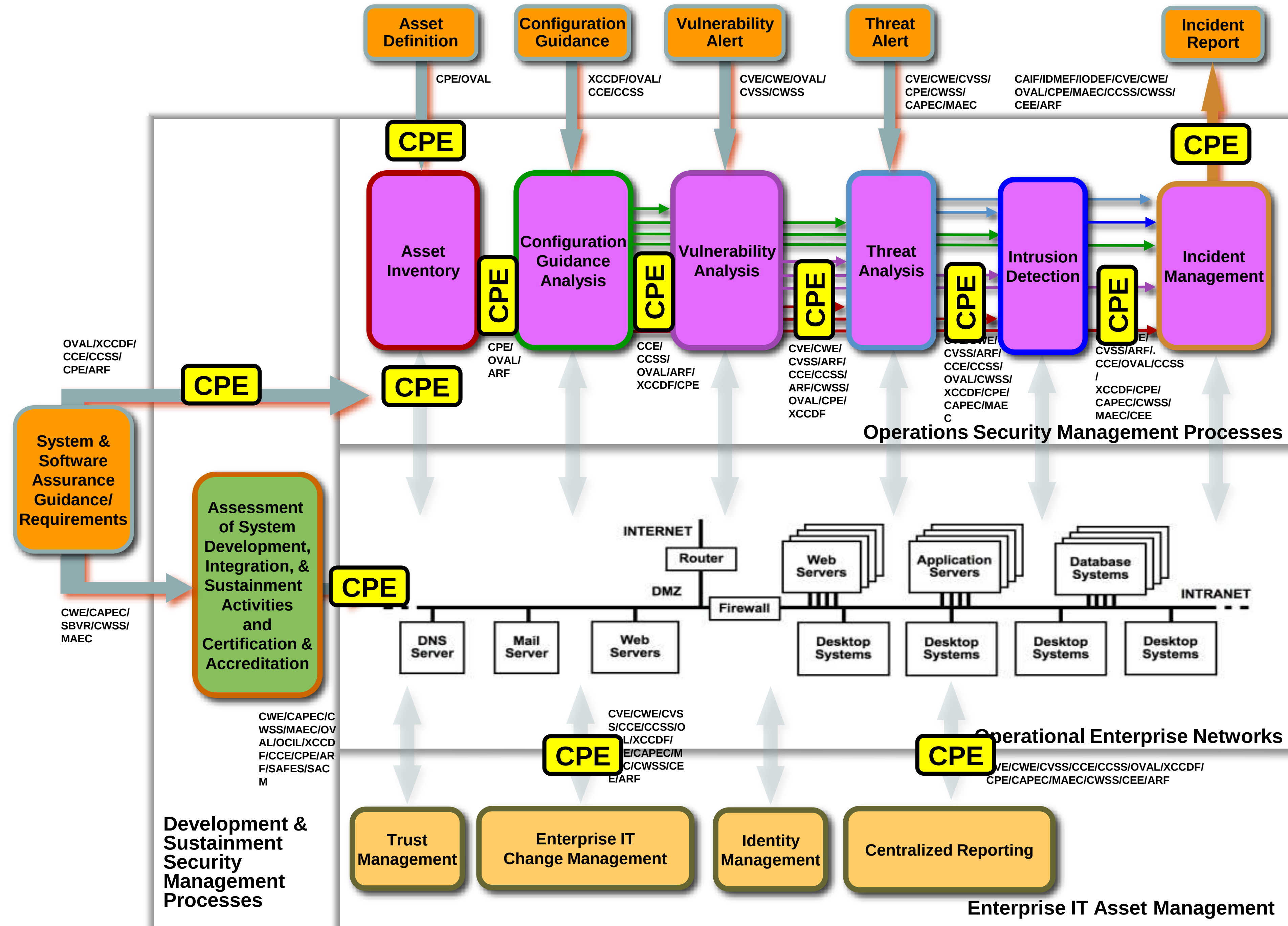
Knowledge Repositories

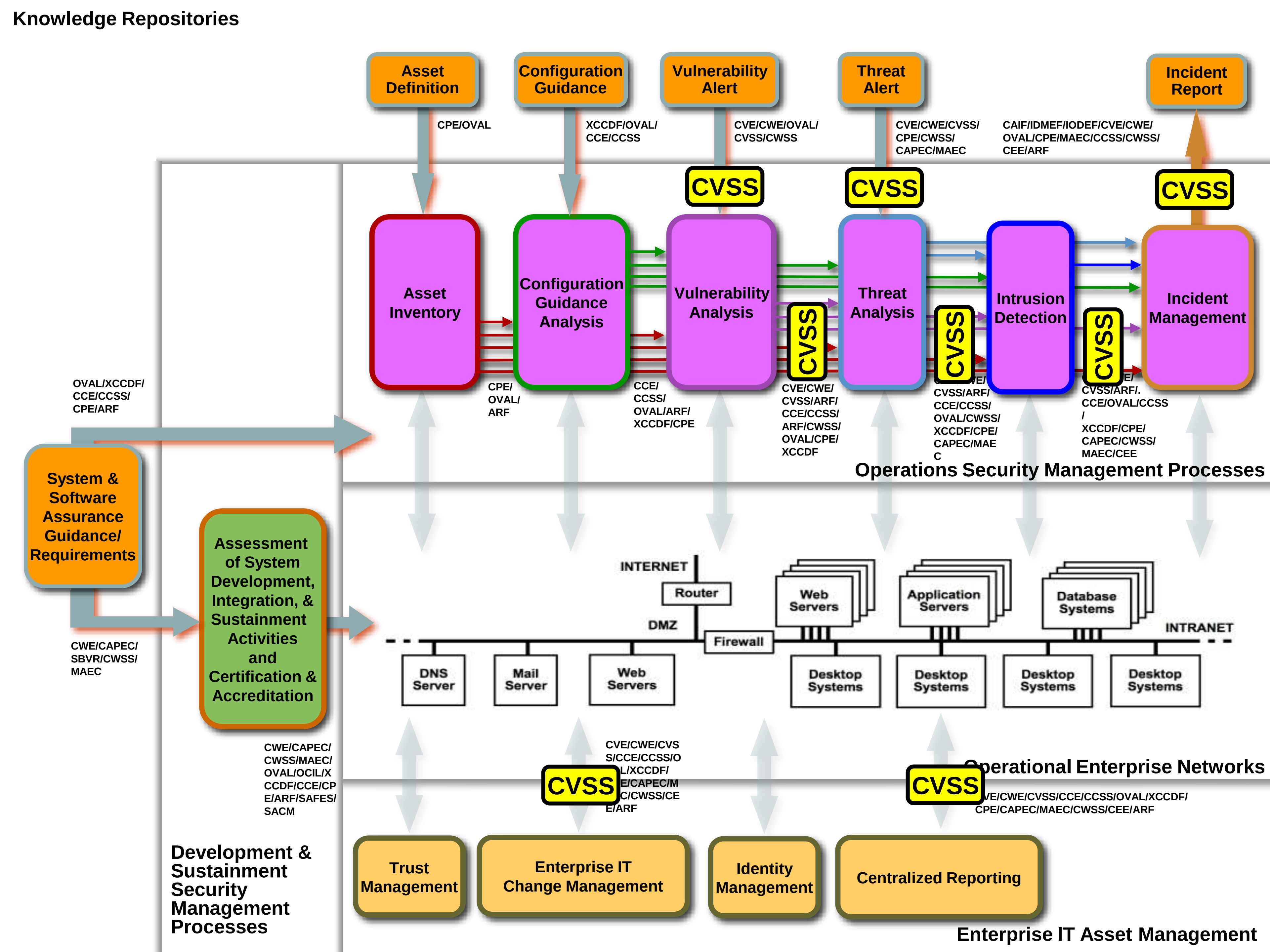


Knowledge Repositories



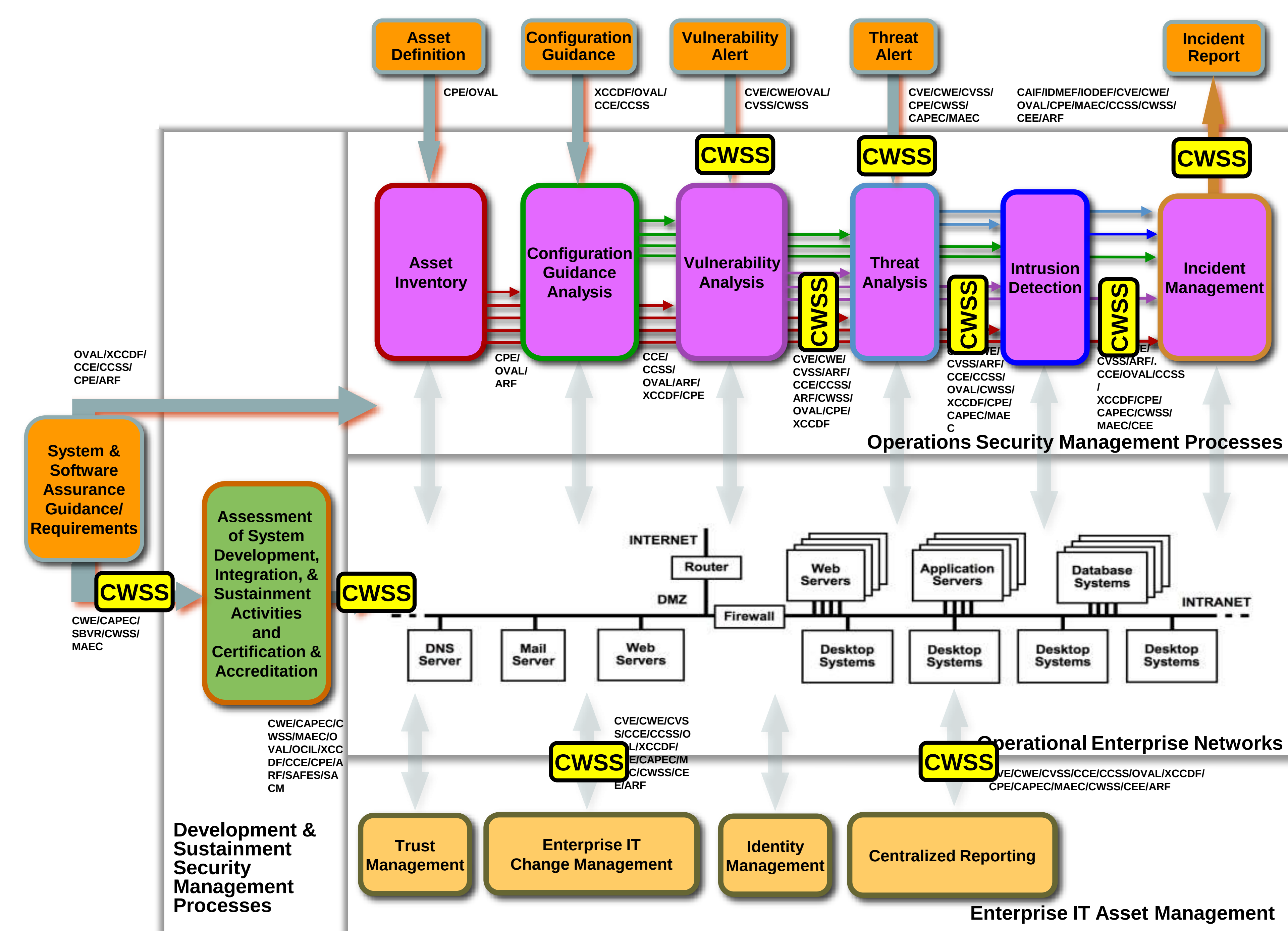
Knowledge Repositories



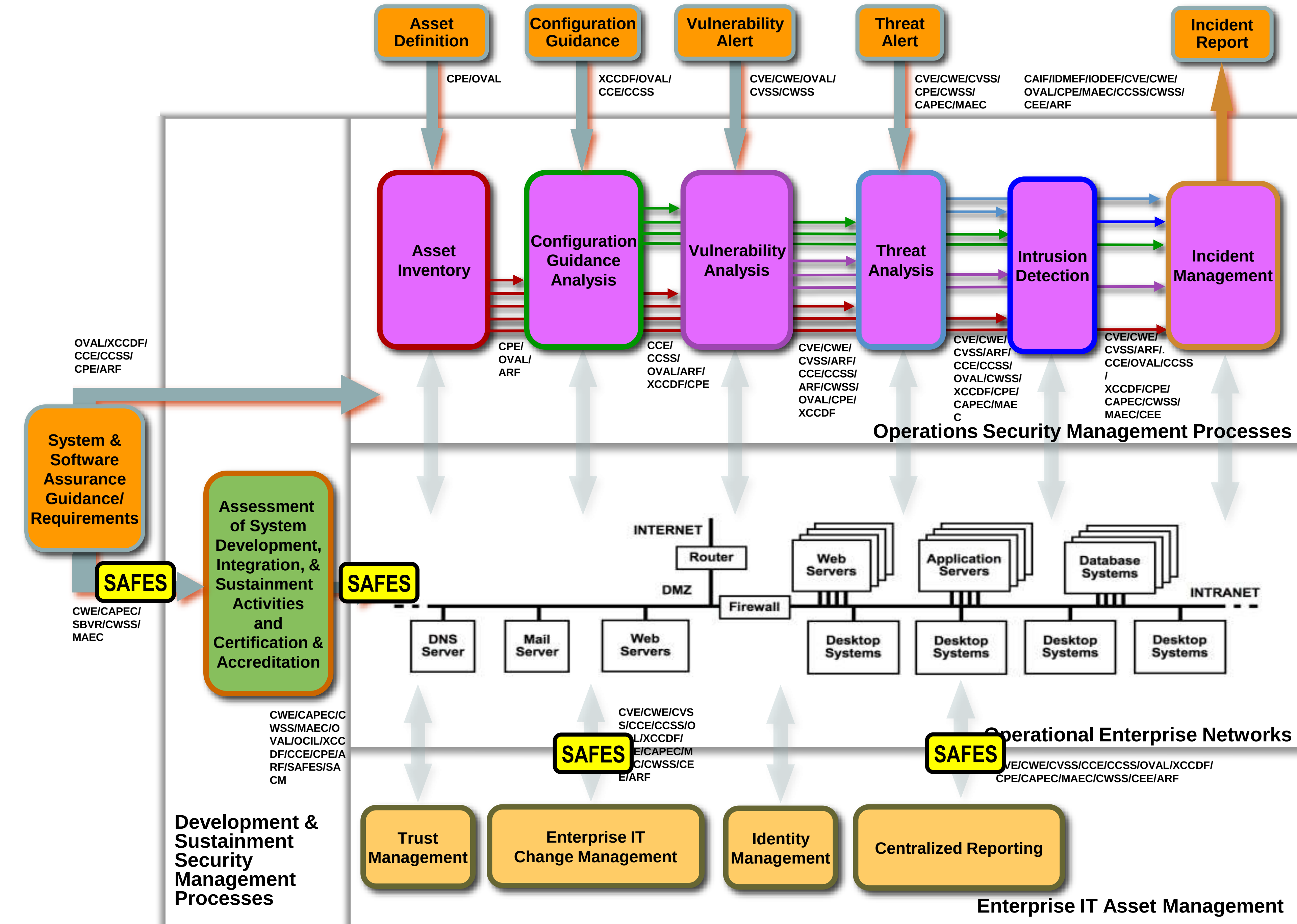


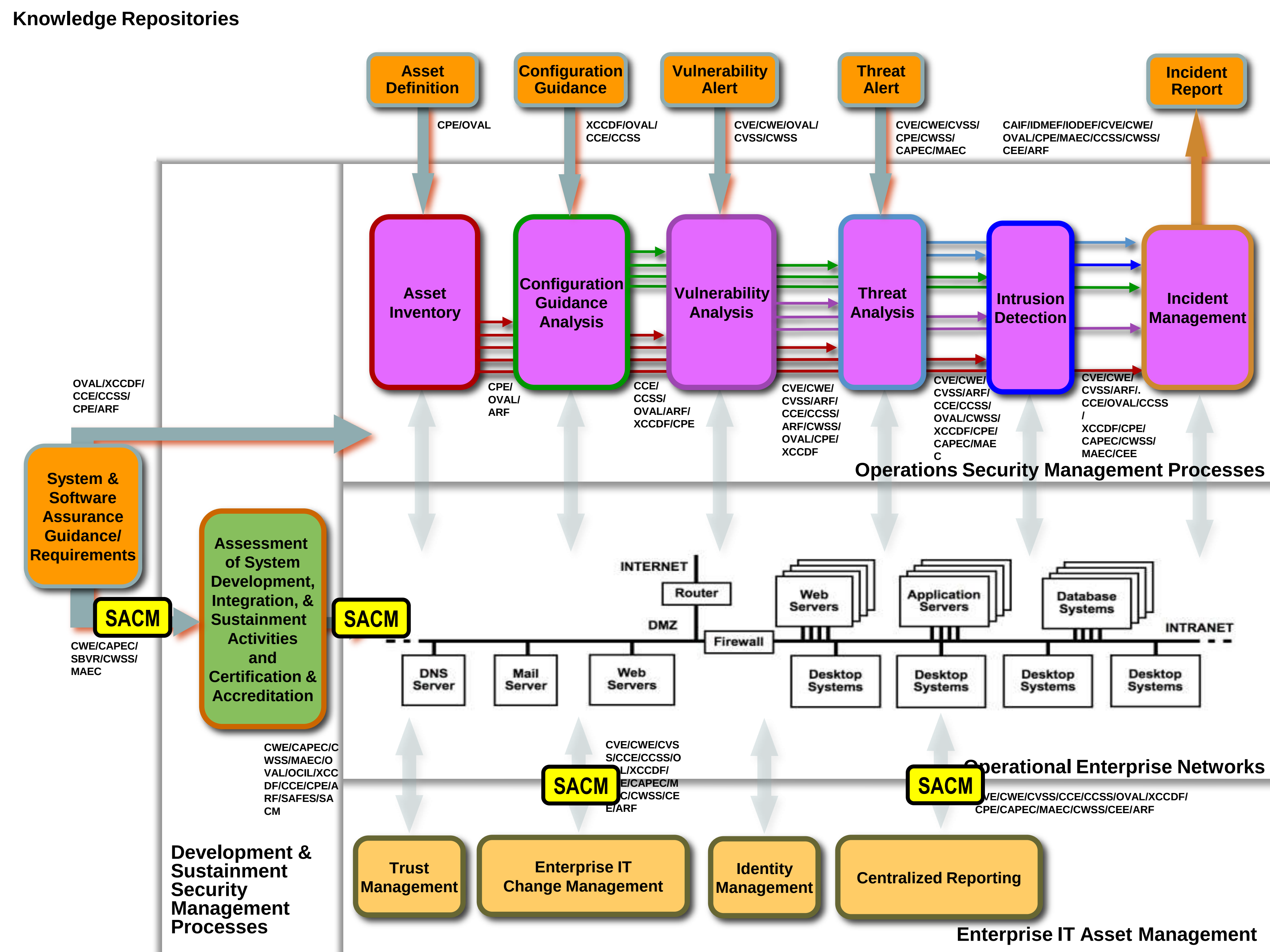
Knowledge Repositories
Asset Definition
Configuration Guidance
Vulnerability Alert
Threat Alert
Incident Report
CPE/OVAL
XCCDF/OVAL/ CCE/CCSS
CVE/CWE/OVAL/ CVSS/CWSS
CVE/CWE/CVSS/ CPE/CWSS/ CAPEC/MAEC
CAIF/IDMEF/IODEF/CVE/CWE/ OVAL/CPE/MAEC/CCSS/CWSS/ CEE/ARF
CVSS
CVSS
CVSS
Asset Inventory
Configuration Guidance Analysis
Vulnerability Analysis
Threat Analysis
Intrusion Detection
Incident Management
CVSS
CVSS
CVSS
CPE/ OVAL/ ARF
CCE/ CCSS/ OVAL/ARF/ XCCDF/CPE
CVE/CWE/ CVSS/ARF/ CCE/CCSS/ ARF/CWSS/ OVAL/CPE/ XCCDF
CVSS/ARF/ CCE/CCSS/ OVAL/CWSS/ XCCDF/CPE/ CAPEC/MAE C
CVSS/ARF/. CCE/OVAL/CCSS / XCCDF/CPE/ CAPEC/CWSS/ MAEC/CEE
OVAL/XCCDF/ CCE/CCSS/ CPE/ARF
System & Software Assurance Guidance/ Requirements
Operations Security Management Processes
Assessment of System Development, Integration, & Sustainment Activities and Certification & Accreditation
CWE/CAPEC/ SBVR/CWSS/ MAEC
INTERNET
Router
DMZ
Firewall
Web Servers
Application Servers
Database Systems
INTRANET
DNS Server
Mail Server
Web Servers
Desktop Systems
Desktop Systems
Desktop Systems
Desktop Systems
CWE/CAPEC/ CWSS/MAEC/ OVAL/OCIL/X CCDF/CCE/CP E/ARF/SAFES/ SACM
CVSS
CVE/CWE/CVS S/CCE/CCSS/O L/XCCDF/ E/CAPEC/M C/CWSS/CE E/ARF
Operational Enterprise Networks
CVSS
VE/CWE/CVSS/CCE/CCSS/OVAL/XCCDF/ CPE/CAPEC/MAEC/CWSS/CEE/ARF
Development & Sustainment Security Management Processes
Trust Management
Enterprise IT Change Management
Identity Management
Centralized Reporting
Enterprise IT Asset Management

Knowledge Repositories

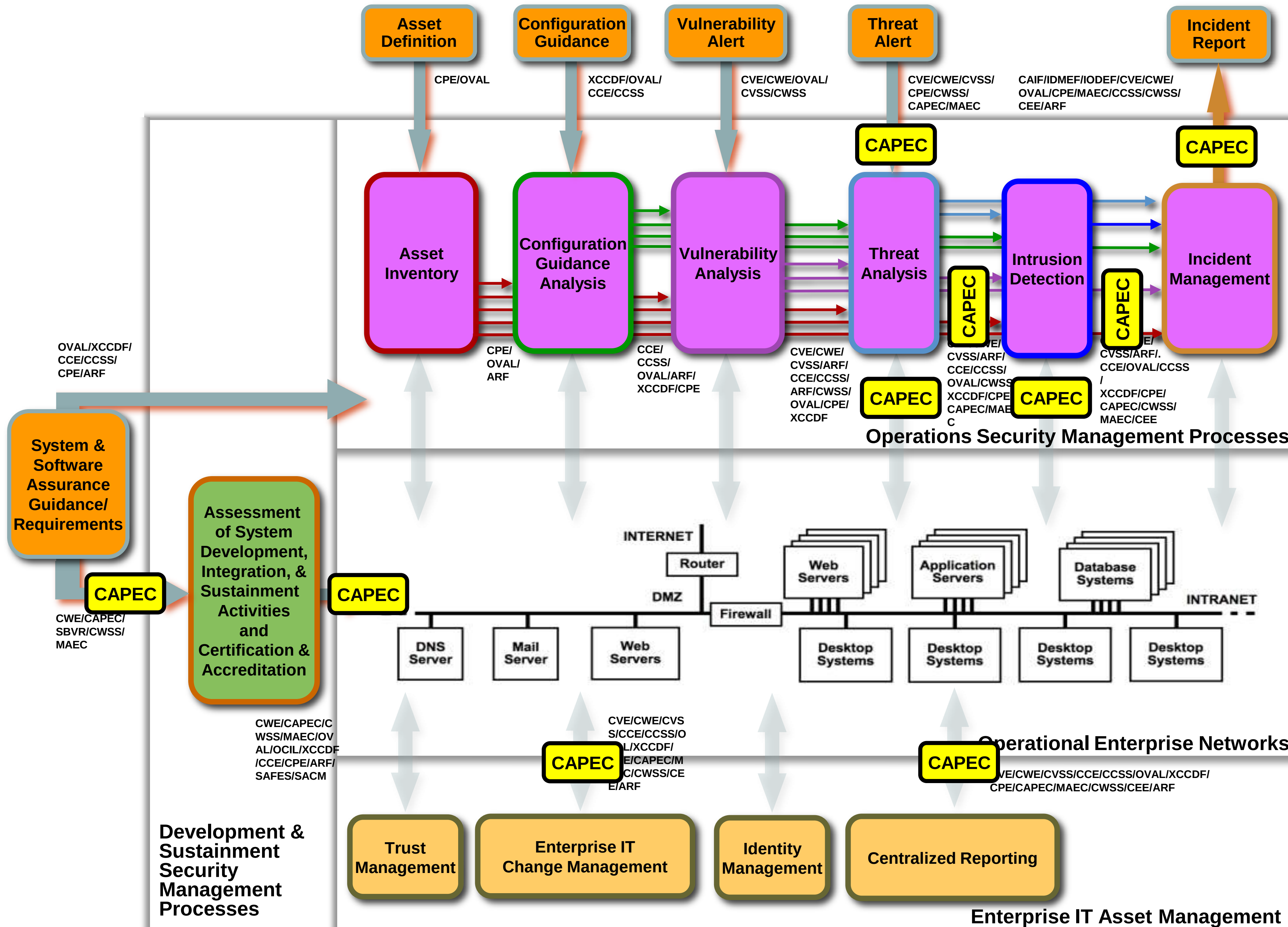


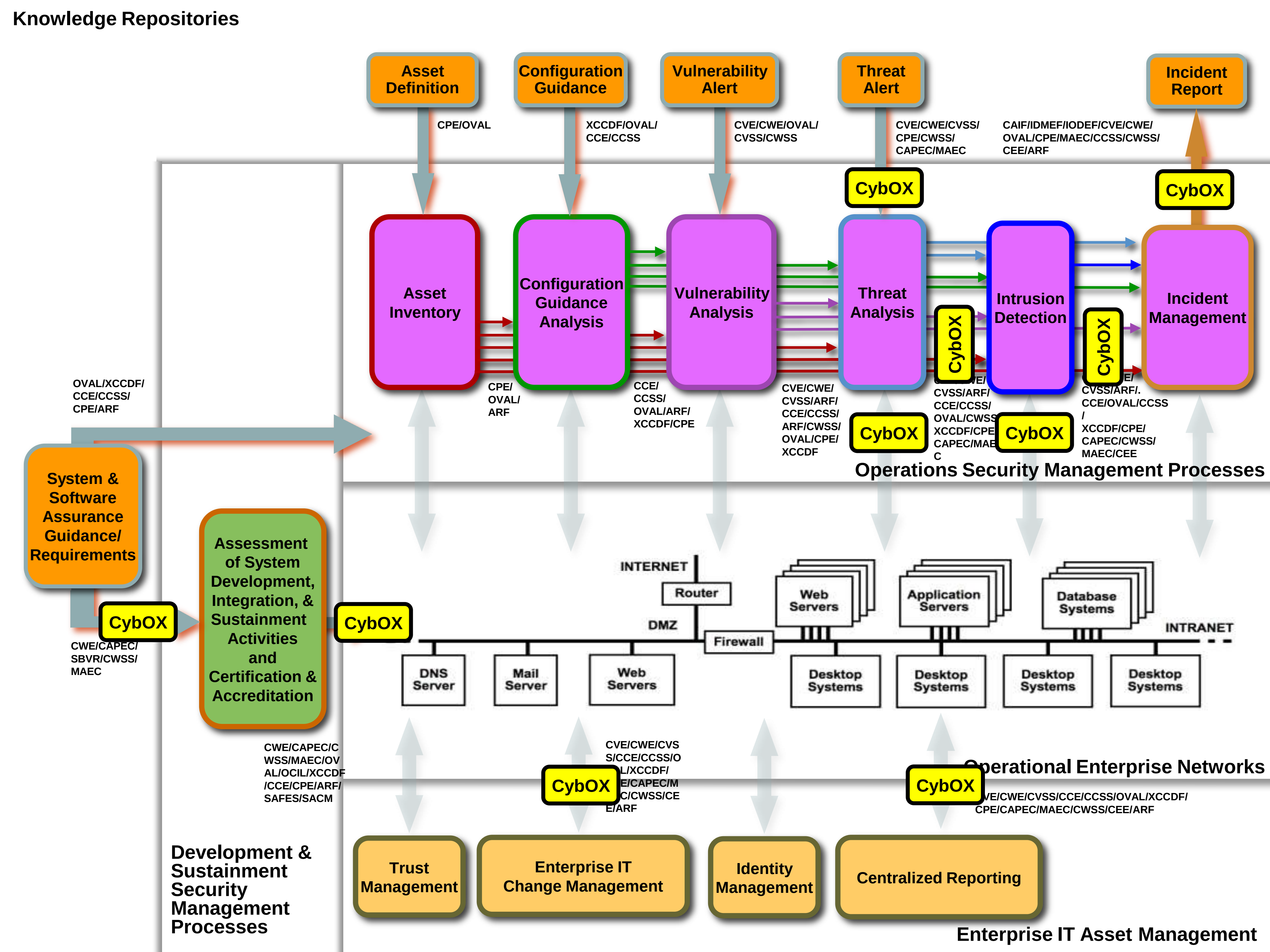
Knowledge Repositories




Knowledge Repositories
Asset Definition
Configuration Guidance
Vulnerability Alert
Threat Alert
Incident Report
CPE/OVAL
XCCDF/OVAL/ CCE/CCSS
CVE/CWE/OVAL/ CVSS/CWSS
CVE/CWE/CVSS/ CPE/CWSS/ CAPEC/MAEC
CAIF/IDMEF/IODEF/CVE/CWE/ OVAL/CPE/MAEC/CCSS/CWSS/ CEE/ARF
Asset Inventory
Configuration Guidance Analysis
Vulnerability Analysis
Threat Analysis
Intrusion Detection
Incident Management
OVAL/XCCDF/ CCE/CCSS/ CPE/ARF
CPE/ OVAL/ ARF
CCE/ CCSS/ OVAL/ARF/ XCCDF/CPE
CVE/CWE/ CVSS/ARF/ CCE/CCSS/ ARF/CWSS/ OVAL/CPE/ XCCDF
CVE/CWE/ CVSS/ARF/ CCE/CCSS/ OVAL/CWSS/ XCCDF/CPE/ CAPEC/MAE C
CVE/CWE/ CVSS/ARF/. CCE/OVAL/CCSS / XCCDF/CPE/ CAPEC/CWSS/ MAEC/CEE
Operations Security Management Processes
System & Software Assurance Guidance/ Requirements
SACM
CWE/CAPEC/ SBVR/CWSS/ MAEC
Assessment of System Development, Integration, & Sustainment Activities and Certification & Accreditation
SACM
INTERNET
Router
DMZ
Firewall
Web Servers
Application Servers
Database Systems
INTRANET
DNS Server
Mail Server
Web Servers
Desktop Systems
Desktop Systems
Desktop Systems
Desktop Systems
CWE/CAPEC/C WSS/MAEC/O VAL/OCIL/XCC DF/CCE/CPE/A RF/SAFES/SA CM
SACM
CVE/CWE/CVS S/CCE/CCSS/O L/XCCDF/ E/CAPEC/M C/CWSS/CE E/ARF
Operational Enterprise Networks
SACM
VE/CWE/CVSS/CCE/CCSS/OVAL/XCCDF/ CPE/CAPEC/MAEC/CWSS/CEE/ARF
Development & Sustainment Security Management Processes
Trust Management
Enterprise IT Change Management
Identity Management
Centralized Reporting
Enterprise IT Asset Management

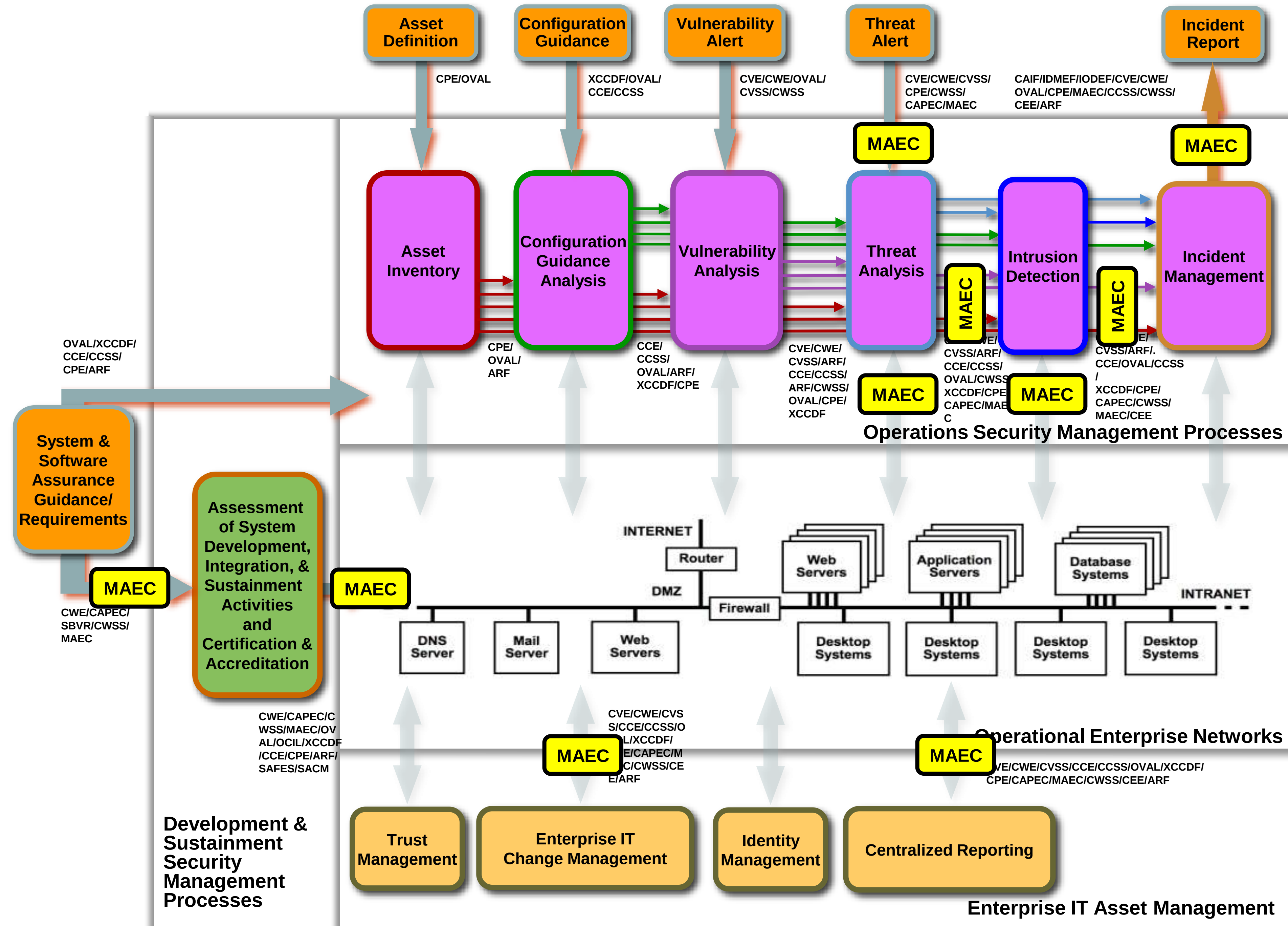
Knowledge Repositories



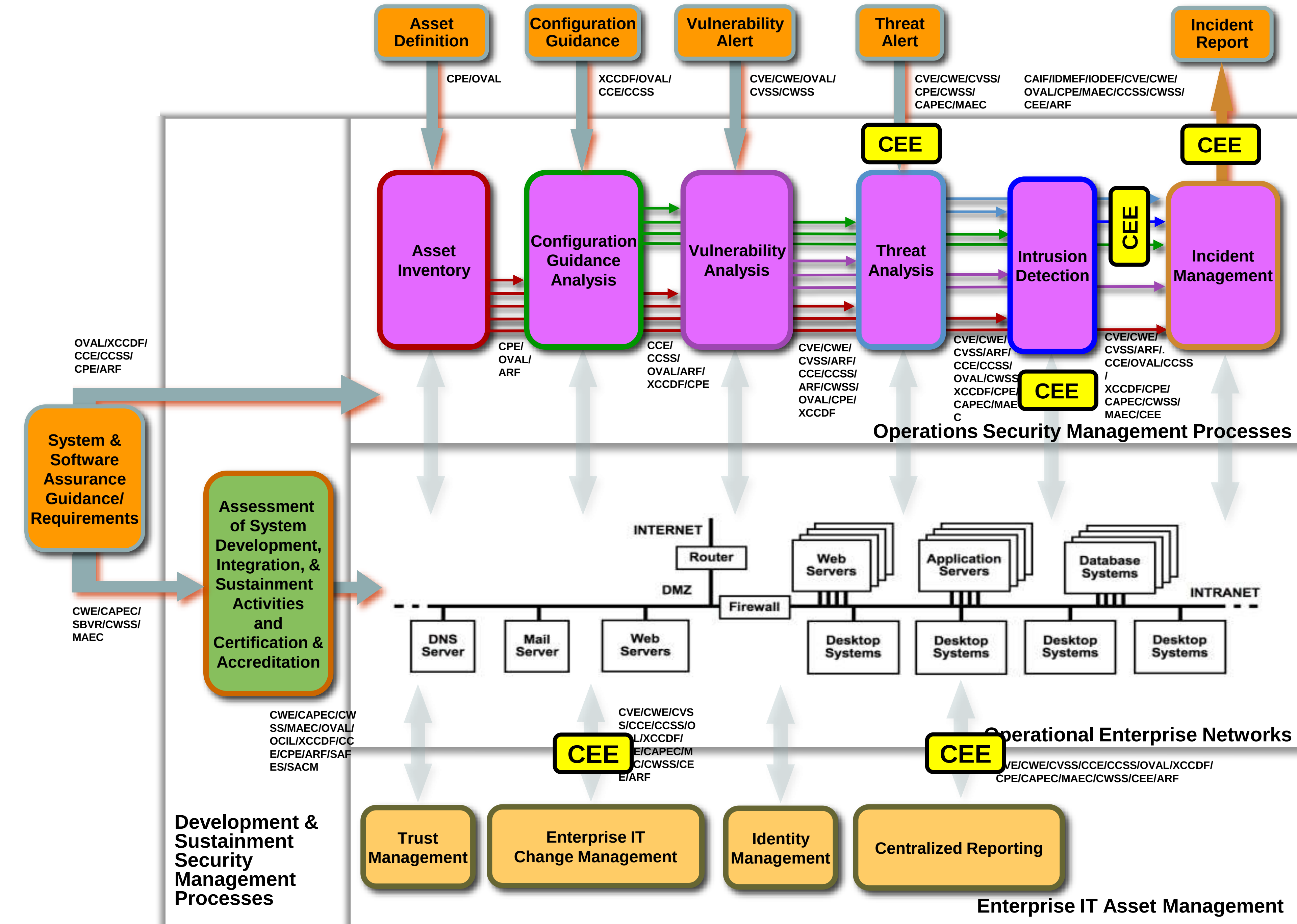


Knowledge Repositories
Asset Definition
Configuration Guidance
Vulnerability Alert
Threat Alert
Incident Report
CPE/OVAL
XCCDF/OVAL/ CCE/CCSS
CVE/CWE/OVAL/ CVSS/CWSS
CVE/CWE/CVSS/ CPE/CWSS/ CAPEC/MAEC
CAIF/IDMEF/IODEF/CVE/CWE/ OVAL/CPE/MAEC/CCSS/CWSS/ CEE/ARF
CybOX
CybOX
Asset Inventory
Configuration Guidance Analysis
Vulnerability Analysis
Threat Analysis
Intrusion Detection
Incident Management
CybOX
CybOX
CPE/ OVAL/ ARF
CCE/ CCSS/ OVAL/ARF/ XCCDF/CPE
CVE/CWE/ CVSS/ARF/ CCE/CCSS/ ARF/CWSS/ OVAL/CPE/ XCCDF
CVSS/ARF/ CCE/CCSS/ OVAL/CWSS XCCDF/CPE CAPEC/MAE C
CVSS/ARF/. CCE/OVAL/CCSS / XCCDF/CPE/ CAPEC/CWSS/ MAEC/CEE
CybOX
CybOX
OVAL/XCCDF/ CCE/CCSS/ CPE/ARF
Operations Security Management Processes
System & Software Assurance Guidance/ Requirements
CybOX
CWE/CAPEC/ SBVR/CWSS/ MAEC
Assessment of System Development, Integration, & Sustainment Activities and Certification & Accreditation
CybOX
INTERNET
Router
DMZ
Firewall
Web Servers
Application Servers
Database Systems
INTRANET
DNS Server
Mail Server
Web Servers
Desktop Systems
Desktop Systems
Desktop Systems
Desktop Systems
CWE/CAPEC/C WSS/MAEC/OV AL/OCIL/XCCDF /CCE/CPE/ARF/ SAFES/SACM
CVE/CWE/CVS S/CCE/CCSS/O L/XCCDF/ E/CAPEC/M C/CWSS/CE E/ARF
CybOX
Operational Enterprise Networks
CybOX
VE/CWE/CVSS/CCE/CCSS/OVAL/XCCDF/ CPE/CAPEC/MAEC/CWSS/CEE/ARF
Development & Sustainment Security Management Processes
Trust Management
Enterprise IT Change Management
Identity Management
Centralized Reporting
Enterprise IT Asset Management

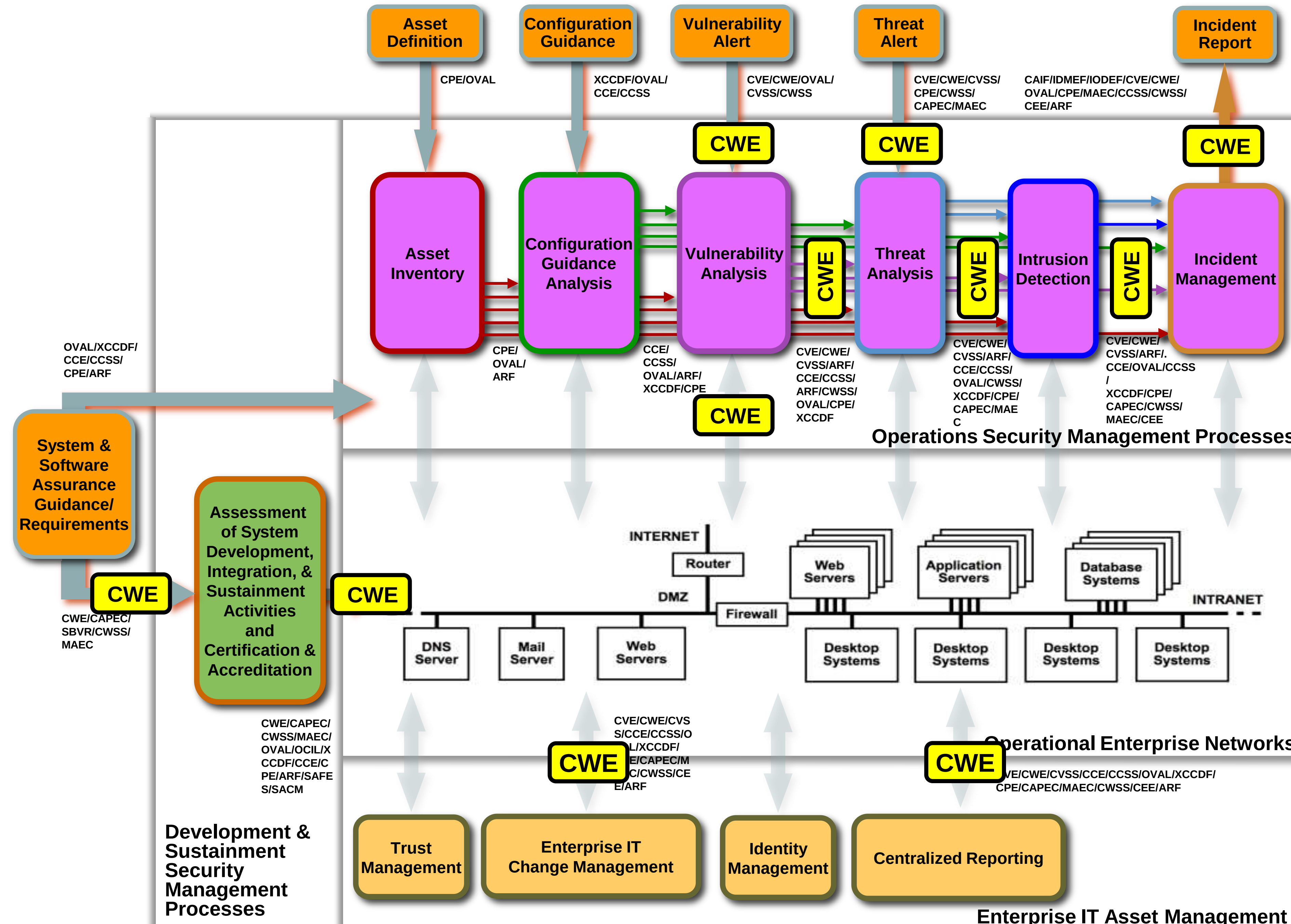
Knowledge Repositories



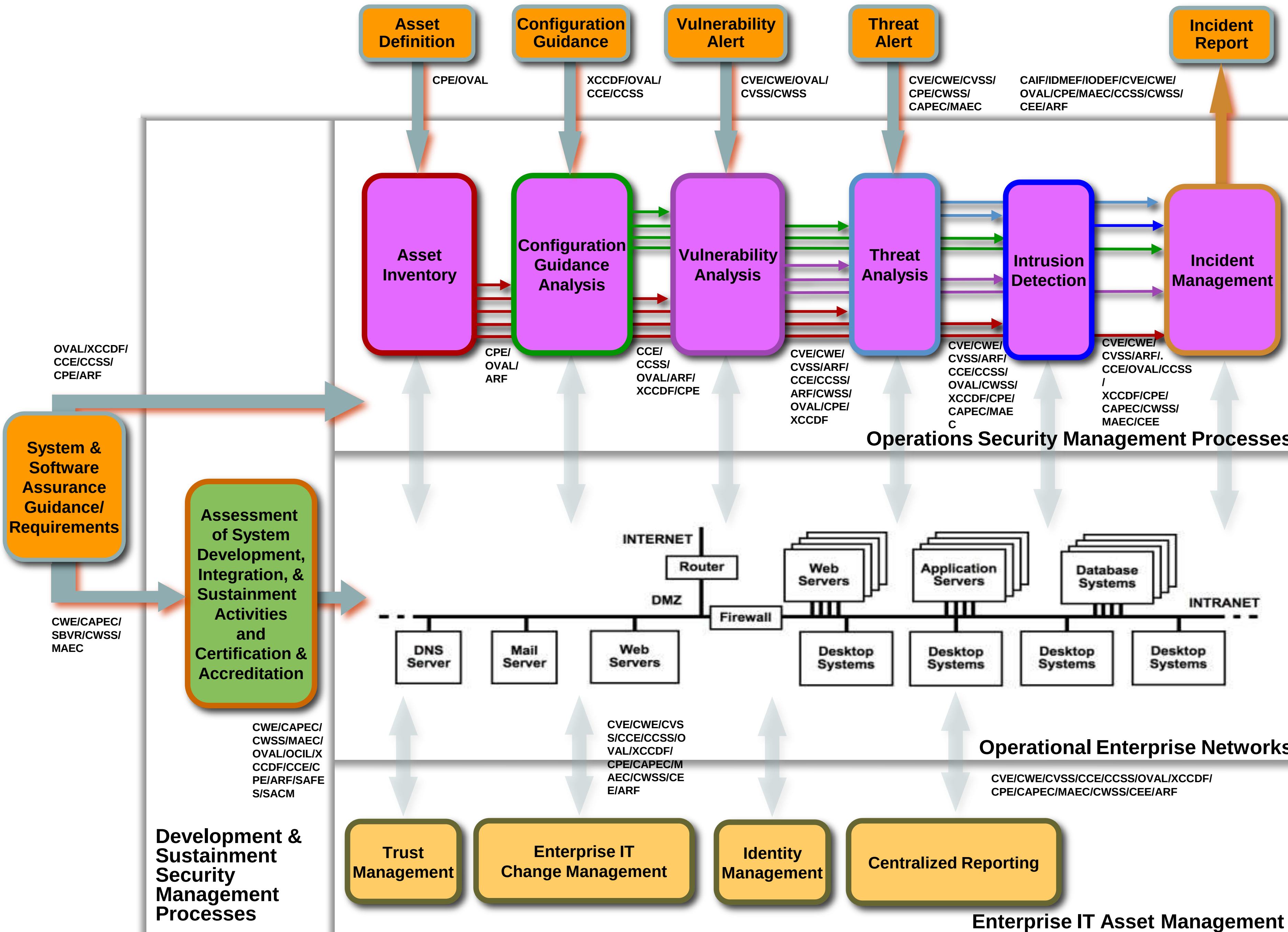
Knowledge Repositories



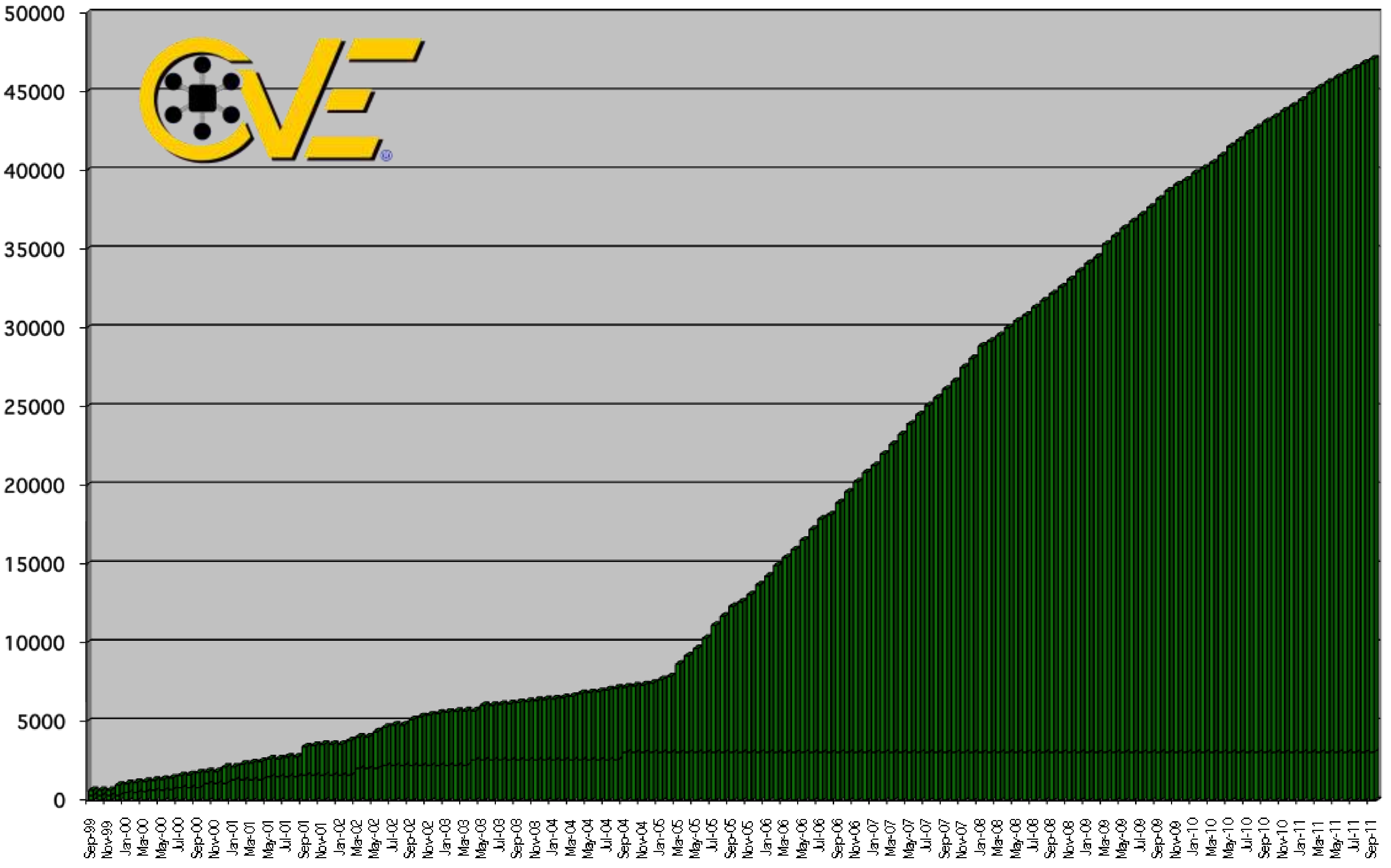
Knowledge Repositories



Knowledge Repositories



CVE 1999 to 2011



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

http://www.microsoft.com/technet/security/Bulletin/MS10-071.mspx

Google

Click Here to Install Silverlight

United States Change | All Microsoft Sites

Microsoft | TechNet

Search Microsoft.com bing Web

TechNet Home | TechCenters | Downloads | TechNet Program | Subscriptions | Security Bulletins | Archive

Search for

Go

- TechNet Security
- Security Bulletin Search
- Library
- Learn
- Downloads
- Support

TechNet Home > TechNet Security > Bulletins

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.

↑ Top of section

⊞ **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**

Mozilla Firefox

http://www.oracle.com/technetwork/topics/security/cpuoct2010-175626.htr

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

CVE#	Component	Protocol	Package and/or Privilege Required	Remote Exploit without Auth.?	CVSS VERSION 2.0 RISK (see Risk Matrix Definitions)							Last Affected Patch set (per Supported Release)	Notes
					Base Score	Access Vector	Access Complexity	Authen-tication	Confiden-tiality	Integrity	Avail-ability		
CVE-2010-2390 (Oracle Enterprise Manager Grid Control)	EM Console	HTTP	None	Yes	7.5	Network	Low	None	Partial+	Partial+	Partial+	10.1.0.5, 10.2.0.3	See Note 1
CVE-2010-2419	Java Virtual Machine	Oracle Net	Create Session	No	6.5	Network	Low	Single	Partial+	Partial+	Partial+	10.1.0.5, 10.2.0.4, 11.1.0.7, 11.2.0.1	
CVE-2010-1321	Change Data Capture	Oracle Net	Execute on DBMS_CDC_PUBLISH	No	5.5	Network	Low	Single	Partial+	Partial+	None	-	See Note 2
CVE-2010-2412	OLAP	Oracle Net	Create Session	No	5.5	Network	Low	Single	Partial+	Partial+	None	11.1.0.7	
CVE-2010-2415	Change Data Capture	Oracle Net	Execute on DBMS_CDC_PUBLISH	No	4.9	Network	Medium	Single	Partial+	Partial+	None	10.1.0.5, 10.2.0.4, 11.1.0.7, 11.2.0.1	
CVE-2010-2411	Job Queue	Oracle Net	Execute on SYS.DBMS_IJOB	No	4.6	Network	High	Single	Partial+	Partial+	Partial+	-	See Note 2
CVE-2010-2407	XDK	HTTP	None	Yes	4.3	Network	Medium	None	None	Partial	None	10.1.0.5, 10.2.0.4, 11.1.0.7	
CVE-2010-2391	Core RDBMS	Oracle Net	Create Session	No	3.6	Network	High	Single	Partial	Partial	None	10.1.0.5, 10.2.0.3	
CVE-2010-2389 (Oracle Fusion Middleware)	Perl	Oracle Net	Local Logon	No	1.0	Local	High	Single	None	Partial+	None	-	See Note 2

rhn.redhat.com | Red Hat Support

http://rhn.redhat.com/errata/RHSA-2010-0723.html

redhat

Errata | Log In | About RHN

Important: kernel security and bug fix update

Advisory:	RHSA-2010:0723-1
Type:	Security Advisory
Severity:	Important
Issued on:	2010-09-29
Last updated on:	2010-09-29
Affected Products:	Red Hat Enterprise Linux (v. 5 server) Red Hat Enterprise Linux Desktop (v. 5 client)
OVAL:	com.redhat.rhsa-20100723.xml
CVEs (cve.mitre.org):	CVE-2010-1083 CVE-2010-2492 CVE-2010-2798 CVE-2010-2938 CVE-2010-2942 CVE-2010-2943 CVE-2010-3015

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

http://lists.apple.com/archives/security-announce/2010//Aug/msg00000.ht

Google

Store | Mac | iPod | iPhone | iPad | iTunes | Support | Search

Mailing Lists

Apple Mailing Lists

Search!

Search only in security-announce list

[Date Prev][Date Next][Thread Prev][Thread Next][Date Index][Thread Index]

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>

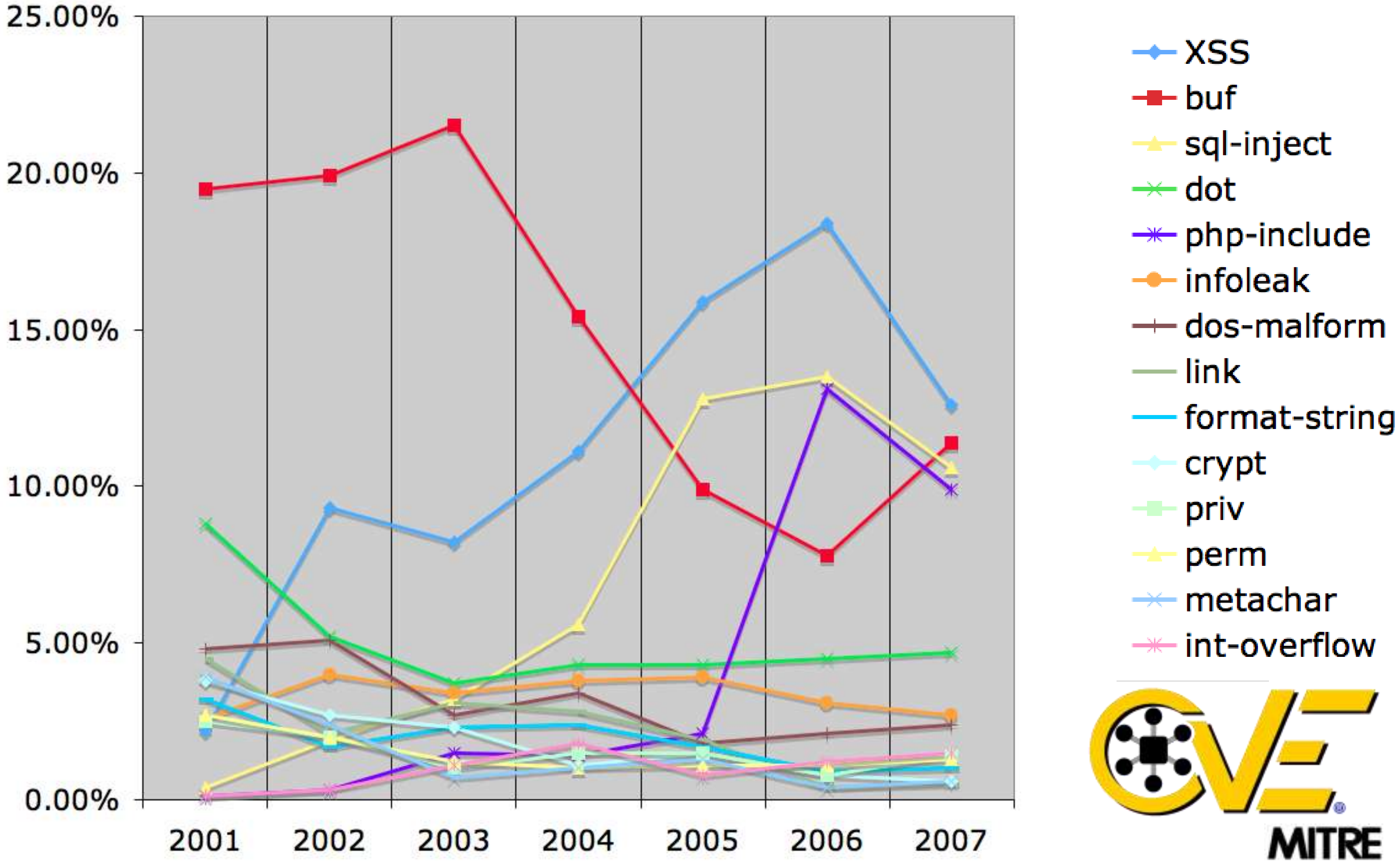
Date: Wed, 11 Aug 2010 12:19:43 -0700

Delivered-to: email@hidden

Delivered-to: email@hidden

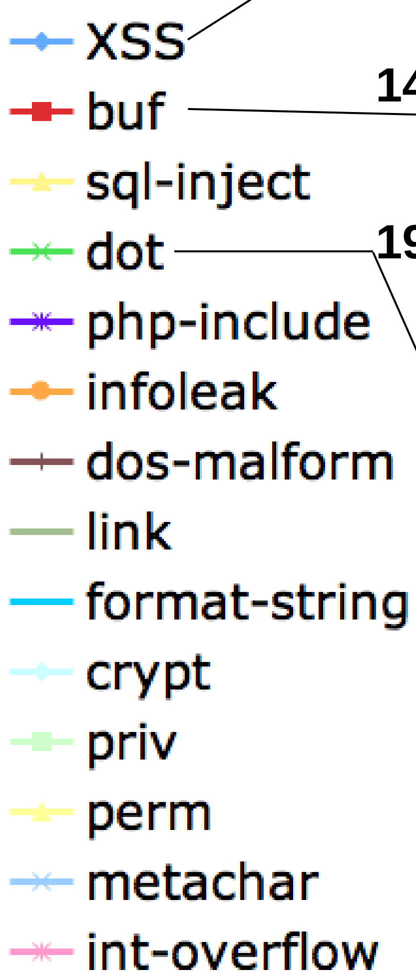
```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

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 4.0 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 opcodes. Viewing a PDF document with maliciously crafted
```

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




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…

Removing and Preventing the Vulnerabilities Requires More Specific Definitions…CWEs



9

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)**

14

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)**

19

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)**

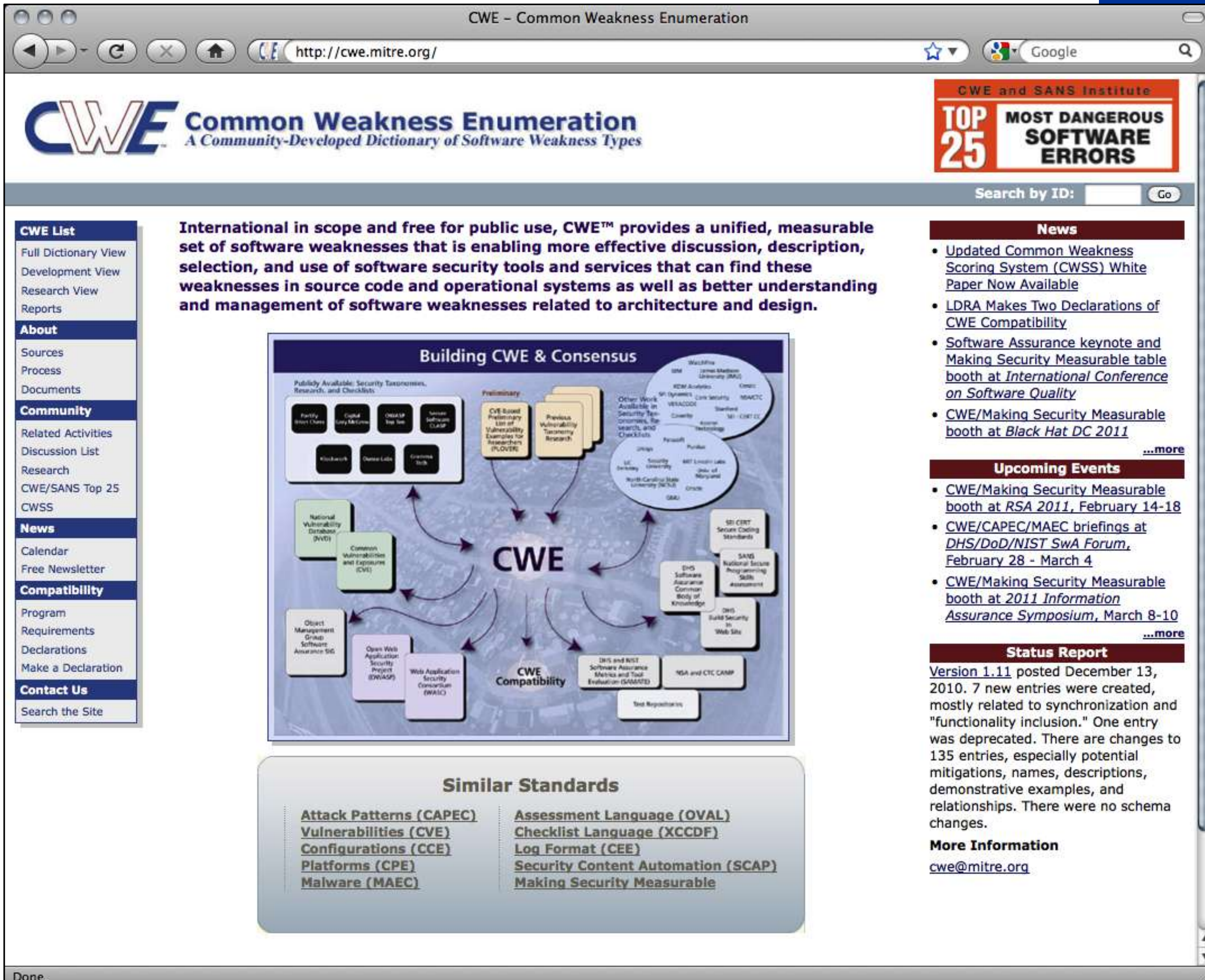
Common Weakness Enumeration (CWE)

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**
  - provides multiple views into CWE dictionary content
  - supports alternate views – developer/researcher/sub-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
- **Where is CWE today?**
  - http://cwe.mitre.org
  - Currently 693 Weaknesses,

    138 Categories and 25 Views

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



CWE – Common Weakness Enumeration

http://cwe.mitre.org/

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

CWE and SANS Institute TOP 25 MOST DANGEROUS SOFTWARE ERRORS

Search by ID: Go

CWE List
- Full Dictionary View
- Development View
- Research View
- Reports

About
- Sources
- Process
- Documents

Community
- Related Activities
- Discussion List
- Research
- CWE/SANS Top 25
- CWSS

News
- Calendar
- Free Newsletter

Compatibility
- Program
- Requirements
- Declarations
- Make a Declaration

Contact Us
- Search the Site

International in scope and free for public use, CWE™ provides a unified, measurable set of software weaknesses that is enabling more effective discussion, description, selection, and use of software security tools and services that can find these weaknesses in source code and operational systems as well as better understanding and management of software weaknesses related to architecture and design.

Building CWE & Consensus

CWE

CWE Compatibility

Similar Standards

- Attack Patterns (CAPEC)
- Vulnerabilities (CVE)
- Configurations (CCE)
- Platforms (CPE)
- Malware (MAEC)
- Assessment Language (OVAL)
- Checklist Language (XCCDF)
- Log Format (CEE)
- Security Content Automation (SCAP)
- Making Security Measurable

News
- Updated Common Weakness Scoring System (CWSS) White Paper Now Available
- LDRA Makes Two Declarations of CWE Compatibility
- Software Assurance keynote and Making Security Measurable table booth at *International Conference on Software Quality*
- CWE/Making Security Measurable booth at *Black Hat DC 2011*

...more

Upcoming Events
- CWE/Making Security Measurable booth at *RSA 2011*, February 14-18
- CWE/CAPEC/MAEC briefings at *DHS/DoD/NIST SwA Forum*, February 28 - March 4
- CWE/Making Security Measurable booth at *2011 Information Assurance Symposium*, March 8-10

...more

Status Report

Version 1.11 posted December 13, 2010. 7 new entries were created, mostly related to synchronization and "functionality inclusion." One entry was deprecated. There are changes to 135 entries, especially potential mitigations, names, descriptions, demonstrative examples, and relationships. There were no schema changes.

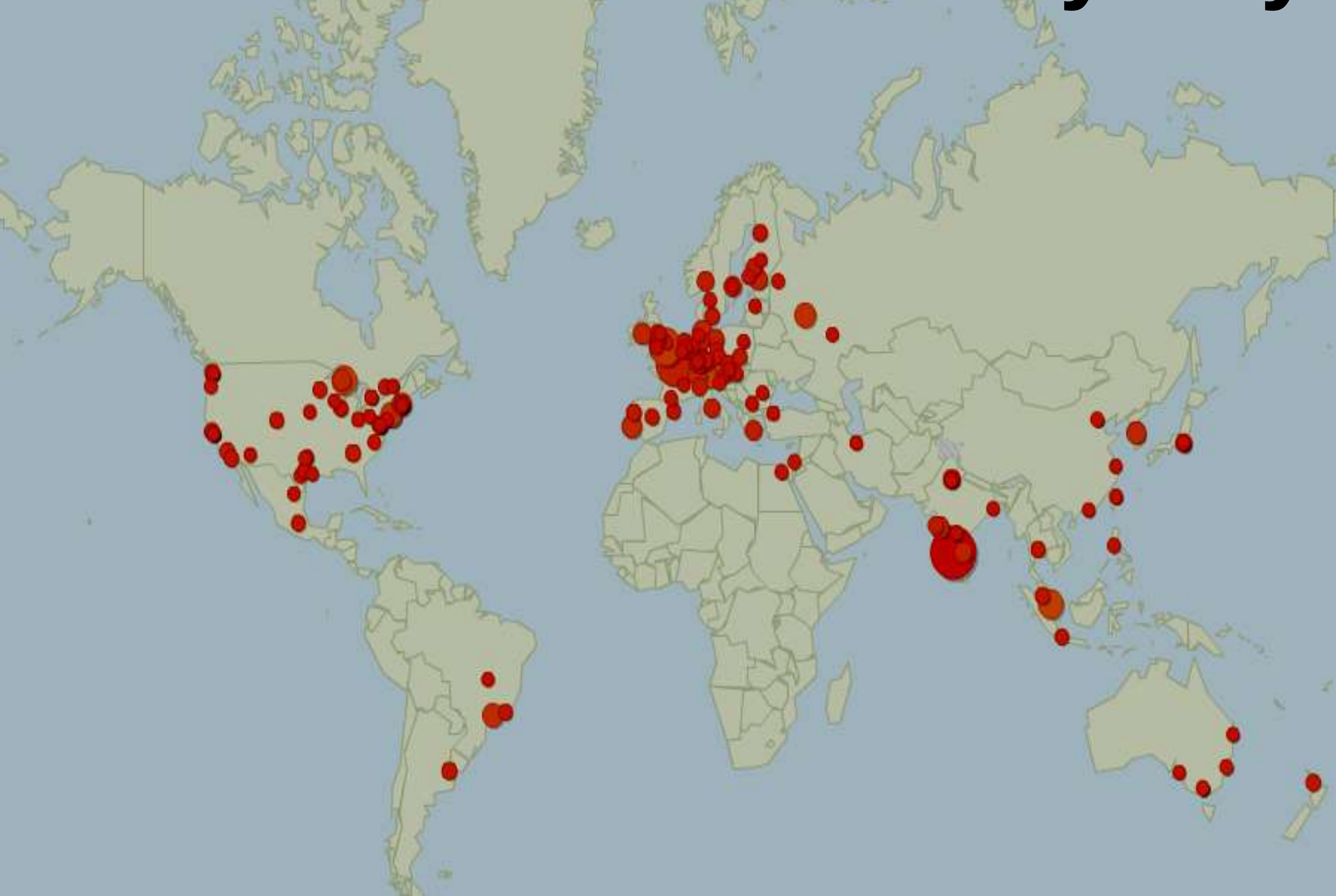
More Information

cwe@mitre.org

Done

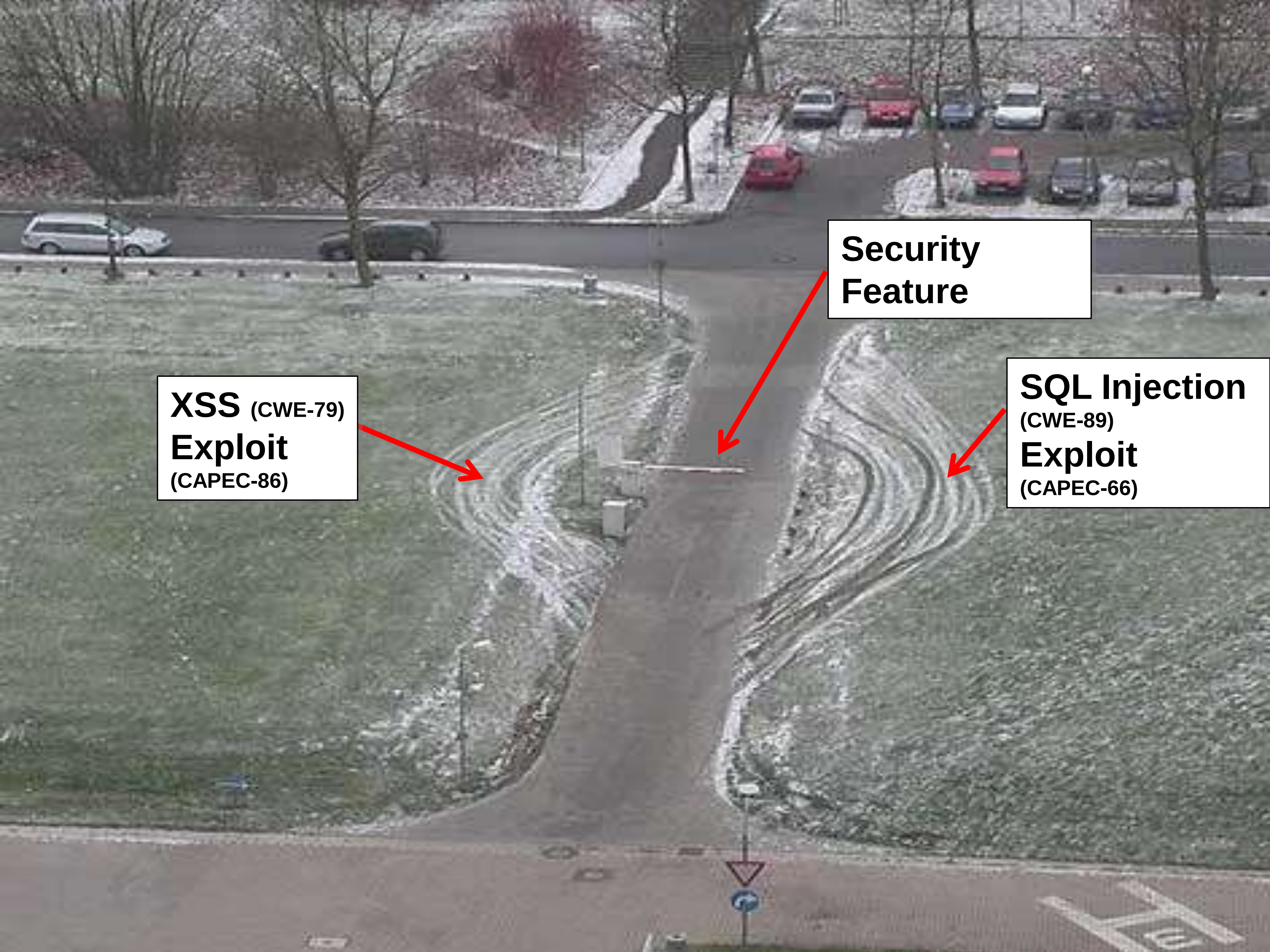
Rank	Score	ID	Name		
[1]	93.8	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	up from 2	+1
[2]	83.3	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	up from 9	+7
[3]	79.0	CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')	same	0
[4]	77.7	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	down from 1	-3
[5]	76.9	CWE-306	Missing Authentication for Critical Function	up from 19	+14
[6]	76.8	CWE-862	Missing Authorization	split of prior #5	-1
[7]	75.0	CWE-798	Use of Hard-coded Credentials	up from 11	+4
[8]	75.0	CWE-311	Missing Encryption of Sensitive Data	up from 10	+2
[9]	74.0	CWE-434	Unrestricted Upload of File with Dangerous Type	down from 8	-1
[10]	73.8	CWE-807	Reliance on Untrusted Inputs in a Security Decision	down from 6	-4
[11]	73.1	CWE-250	Execution with Unnecessary Privileges	new entry	n/a
[12]	70.1	CWE-352	Cross-Site Request Forgery (CSRF)	down from 4	-8
[13]	69.3	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	down from 7	-6
[14]	68.5	CWE-494	Download of Code Without Integrity Check	up from 20	+6
[15]	67.8	CWE-863	Incorrect Authorization	split of prior #5	-10

CWE web site visitors by City



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.


Security Feature
XSS (CWE-79) Exploit (CAPEC-86)
SQL Injection (CWE-89) Exploit (CAPEC-66)

"Know Your Enemy"

- "One who knows the enemy and knows himself will not be endangered in a hundred engagements. One who does not know the enemy but knows himself will sometimes be victorious. Sometimes meet with defeat. One who knows neither the enemy nor himself will invariably be defeated in every engagement."

  - **Chapter 3: "Planning the Attack"**
    - **The Art of War, Sun Tzu**

The Importance of Knowing Your Enemy

- An appropriate defense can only be established if you know how it will be attacked

- Remember!
  - **Software Assurance must assume motivated attackers and not simply passive quality issues**
  - **Attackers are very creative and have powerful tools at their disposal**
  - **Exploring the attacker’s perspective helps to identify and qualify the risk profile of the software**

What are Attack Patterns?

- Blueprint for creating a specific type of attack
- Abstracted common attack approaches from the set of known exploits
- Capture the attacker’s perspective to aid software developers, acquirers and operators in improving the assurance profile of their software

Leveraging Attack Patterns Throughout the Software Lifecycle

- Guide definition of appropriate policies
- Guide creation of appropriate security requirements (positive and negative)
- Provide context for architectural risk analysis
- Guide risk-driven secure code review
- Provide context for appropriate security testing
- Provide a bridge between secure development and secure operations

Common Attack Pattern Enumeration and Classification (CAPEC)

- Community effort targeted at:
  - Standardizing the capture and description of attack patterns
  - Collecting known attack patterns into an integrated enumeration that can be consistently and effectively leveraged by the community
  - Gives you an attacker’s perspective you may not have on your own

- Excellent resource for many key activities
  - Abuse Case development
  - Architecture attack resistance analysis
  - Risk-based security/Red team penetration
  - Whitebox and Blackbox testing correlation
  - Operational observation and correlation

- Where is CAPEC today?
  - http://capec.mitre.org
  - Currently 386 patterns, stubs, named attacks
    68 Categories & 6 Views


CAPEC - Common Attack Pattern Enumeration and Classification (CAPEC)
http://capec.mitre.org/
Google
CAPEC
Common Attack Pattern Enumeration and Classification
A Community Knowledge Resource for Building Secure Software
Search by ID:
Go
CAPEC List
Full CAPEC Dictionary
Methods of Attack View
Reports
About CAPEC
Documents
Resources
Community
Related Activities
Collaboration List
News & Events
Calendar
Free Newsletter
Contact Us
Search the Site
Building software with an adequate level of security assurance for its mission becomes more and more challenging every day as the size, complexity, and tempo of software creation increases and the number and the skill level of attackers continues to grow. These factors each exacerbate the issue that, to build secure software, builders must ensure that they have protected every relevant potential vulnerability; yet, to attack software, attackers often have to find and exploit only a single exposed vulnerability. To identify and mitigate relevant vulnerabilities in software, the development community needs more than just good software engineering and analytical practices, a solid grasp of software security features, and a powerful set of tools. All of these things are necessary but not sufficient. To be effective, the community needs to think outside of the box and to have a firm grasp of the attacker's perspective and the approaches used to exploit software.
Attack patterns are a powerful mechanism to capture and communicate the attacker's perspective. They are descriptions of common methods for exploiting software. They derive from the concept of design patterns applied in a destructive rather than constructive context and are generated from in-depth analysis of specific real-world exploit examples.
To assist in enhancing security throughout the software development lifecycle, and to support the needs of developers, testers and educators, the **Common Attack Pattern Enumeration and Classification (CAPEC)** is sponsored by the Department of Homeland Security as part of the Software Assurance strategic initiative of the National Cyber Security Division. The objective of this effort is to provide a publicly available catalog of attack patterns along with a comprehensive schema and classification taxonomy. This site now contains the initial set of content and will continue to evolve with public participation and contributions to form a standard mechanism for identifying, collecting, refining, and sharing attack patterns among the software community.
Release 1.6 Available
Page Last Updated: February 07, 2011
MITRE
CAPEC is a Software Assurance strategic initiative co-sponsored by the National Cyber Security Division of the U.S. Department of Homeland Security.
This Web site is sponsored and managed by The MITRE Corporation to enable stakeholder collaboration.
Copyright 2011, The MITRE Corporation. CAPEC and the CAPEC logo are trademarks of The MITRE Corporation.
Contact capec@mitre.org for more information.
Privacy policy
Terms of use
Contact us
Making Security Measurable
Done


http://cwe.mitre.org
CWE
CAPEC
http://capec.mitre.org

What do Attack Patterns Look Like?

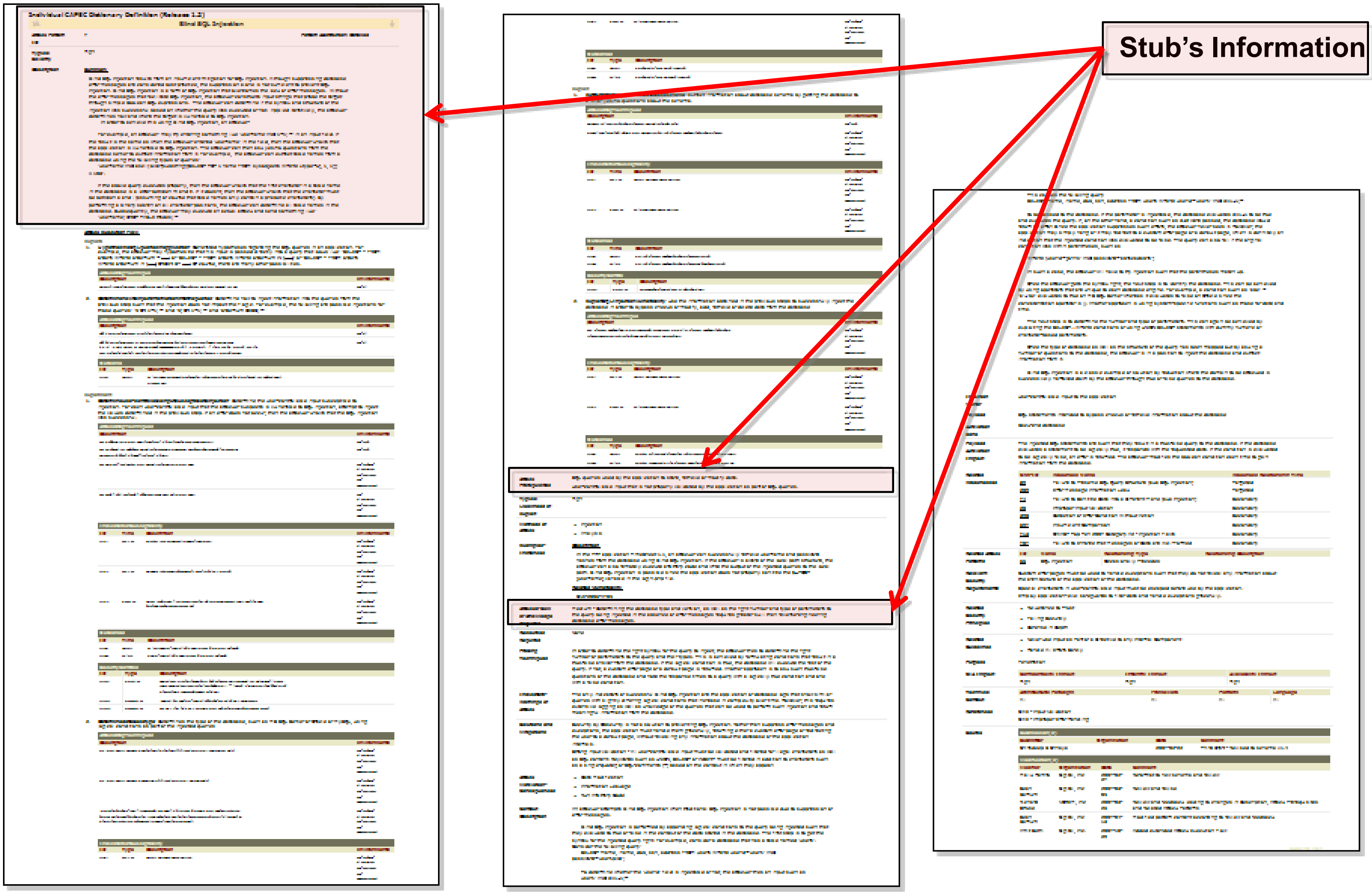
- Primary Schema Elements
  - **Identifying Information**
    - Attack Pattern ID
    - Attack Pattern Name
  - **Describing Information**
    - Description
    - Related Weaknesses
    - Related Vulnerabilities
    - Method of Attack
    - Examples-Instances
    - References
  - **Prescribing Information**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

Attack Pattern Description Schema Formalization

Description

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

Complete CAPEC Entry Information



CAPEC Current Content
(15 Major Categories)

1000 - Mechanism of Attack

- **Data Leakage Attacks - (118)**
- **Resource Depletion - (119)**
- **Injection (Injecting Control Plane content through the Data Plane) - (152)**
- **Spoofing - (156)**
- **Time and State Attacks - (172)**
- **Abuse of Functionality - (210)**
- **Exploitation of Authentication - (225)**
- **Probabilistic Techniques - (223)**
- **Exploitation of Privilege/Trust - (232)**
- **Data Structure Attacks - (255)**
- **Resource Manipulation - (262)**
- **Physical Security Attacks (436)**
- **Network Reconnaissance - *(286)***
- **Social Engineering Attacks (403)**
- **Supply Chain Attacks (437)**

CAPEC Current Content (Which Expand to…)

- 1000 - Mechanism of Attack
  - Data Leakage Attacks - (118)
    - Data Excavation Attacks - (116)
    - Data Interception Attacks - (117)
  - Resource Depletion - (119)
    - Violating Implicit Assumptions Regarding XML Content (aka XML Denial of Service (XDoS)) - (82)
    - Resource Depletion through Flooding - (125)
    - Resource Depletion through Allocation - (130)
    - Resource Depletion through Leak - (131)
    - Denial of Service through Resource Depletion - (227)
  - Injection (Injecting Control Plane content through the Data Plane) - (152)
    - Remote Code Inclusion - (253)
    - Analog In-band Switching Signals (aka Blue Boxing) - (5)
    - SQL Injection - (66)
    - Email Injection - (134)
    - Format String Injection - (135)
    - LDAP Injection - (136)
    - Parameter Injection - (137)
    - Reflection Injection - (138)
    - Code Inclusion - (175)
    - Resource Injection - (240)
    - Script Injection - (242)
    - Command Injection - (248)
    - Character Injection - (249)
    - XML Injection - (250)
    - DTD Injection in a SOAP Message - (254)
  - Spoofing - (156)
    - Content Spoofing - (148)
    - Identity Spoofing (Impersonation) - (151)
    - Action Spoofing - (173)
  - Time and State Attacks - (172)
    - Forced Deadlock - (25)
    - Leveraging Race Conditions - (26)
    - Leveraging Time-of-Check and Time-of-Use (TOCTOU) Race Conditions - (29)
    - Manipulating User State - (74)
  - Abuse of Functionality - (210)
    - Functionality Misuse - (212)
    - Abuse of Communication Channels - (216)
    - Forceful Browsing - (87)
    - Passing Local Filenames to Functions That Expect a URL - (48)
    - Probing an Application Through Targeting its Error Reporting - (54)
    - WSDL Scanning - (95)
    - API Abuse/Misuse - (113)
    - Try All Common Application Switches and Options - (133)
    - Cache Poisoning - (141)
    - Software Integrity Attacks - (184)
    - Directory Traversal - (213)
    - Analytic Attacks - (281)
  - Probabilistic Techniques - (223)
    - Fuzzing - (28)
    - Manipulating Opaque Client-based Data Tokens - (39)
    - Brute Force - (112)
    - Screen Temporary Files for Sensitive Information - (155)
  - Exploitation of Authentication - (225)
    - Exploitation of Session Variables, Resource IDs and other Trusted Credentials - (21)
    - Authentication Abuse - (114)
    - Authentication Bypass - (115)
  - Exploitation of Privilege/Trust - (232)
    - Privilege Escalation - (233)
    - Exploiting Trust in Client (aka Make the Client Invisible) - (22)
    - Hijacking a Privileged Thread of Execution - (30)
    - Subvert Code-signing Facilities - (68)
    - Target Programs with Elevated Privileges - (69)
    - Exploitation of Authorization - (122)
    - Hijacking a privileged process - (234)
  - Data Structure Attacks - (255)
    - Accessing/Intercepting/Modifying HTTP Cookies - (31)
    - Buffer Attacks - (123)
    - Attack through Shared Data - (124)
    - Integer Attacks - (128)
    - Pointer Attack - (129)
  - Resource Manipulation - (262)
    - Accessing/Intercepting/Modifying HTTP Cookies - (31)
    - Input Data Manipulation - (153)
    - Resource Location Attacks - (154)
    - Infrastructure Manipulation - (161)
    - File Manipulation - (165)
    - Variable Manipulation - (171)
    - Configuration/Environment manipulation - (176)
    - Abuse of transaction data strutcture - (257)
    - Registry Manipulation - (269)
    - Schema Poisoning - (271)
    - Protocol Manipulation - (272)
  - Network Reconnaissance - (286)
    - ICMP Echo Request Ping - (285)
    - TCP SYN Scan - (287)
    - ICMP Echo Request Ping - (288)
    - Infrastructure-based footprinting - (289)
    - Enumerate Mail Exchange (MX) Records - (290)
    - DNS Zone Transfers - (291)
    - Host Discovery - (292)
    - Traceroute Route Enumeration - (293)
    - ICMP Address Mask Request - (294)
    - ICMP Timestamp Request - (295)
    - ICMP Information Request - (296)
    - TCP ACK Ping - (297)
    - UDP Ping - (298)
    - TCP SYN Ping - (299)
    - Port Scanning - (300)
    - TCP Connect Scan - (301)
    - TCP FIN scan - (302)
    - TCP Xmas Scan - (303)
    - TCP Null Scan - (304)
    - TCP ACK Scan - (305)
    - TCP Window Scan - (306)
    - TCP RPC Scan - (307)
    - UDP Scan - (308)

CAPEC Current Content (386 Attacks…)

A Few Key Use Cases for CAPEC in Support of SwA

- Help developers understand weaknesses in their real-world context (how they will be attacked)
- Objectively identify specific attacks under which software must demonstrate resistance, tolerance and resilience for a given level of assurance
- Indirectly scope which weaknesses are relevant for a given threat environment
- Identify relevant mitigations that should be applied as part of policy, requirements, A&D, implementation, test, deployment and operations
- Identify and characterize patterns of attacks for security test case generation
- Identify and characterize threat TTPs for red teaming
- Identify relevant issues for automated tool selection
- Identify and characterize issues for automated tool results analysis

CVE®

CWE™

http://cwe.mitre.org

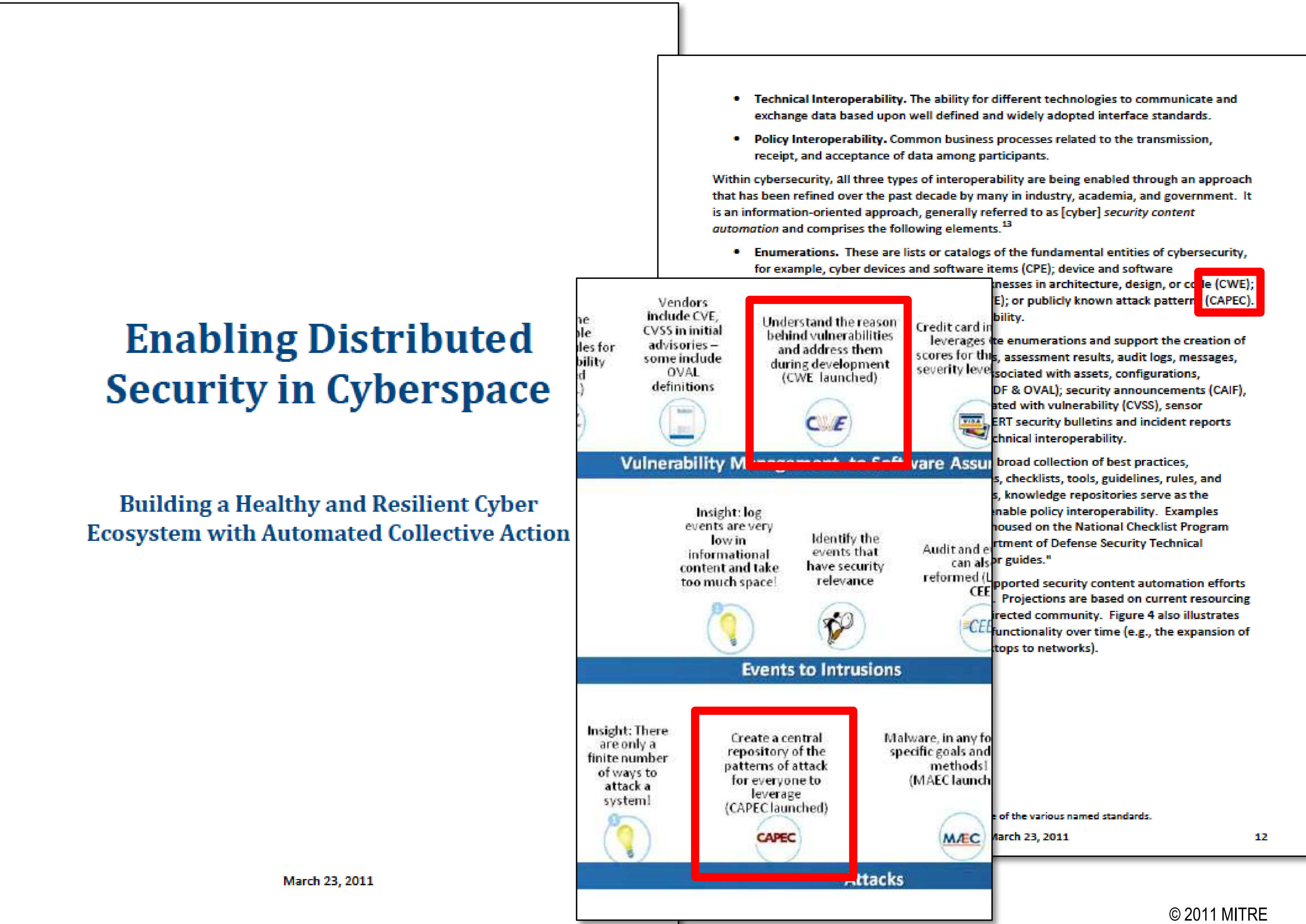
CWE™

CAPEC™

http://capec.mitre.org

CAPEC™

Linkage with Fundamental Changes in Enterprise Security Initiatives



The Security Development Lifecycle : MS08-078 and the SDL

http://blogs.msdn.com/sdl/archive/2008/12/18/ms08-078-and-the-sdl.aspx

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)

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

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

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

Common Weakness Enumeration

A Community-Developed Dictionary of Software Weakness Types

CWE-367: Time-of-check Time-of-use (TOCTOU) Race Condition

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

Weakness ID: 367 *(Weakness Base)*

Status: Incomplete

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

Time of Introduction

- Implementation

Applicable Platforms

Languages

All

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.

IBM Software
Technical White Paper

One way to improve software security is to gain a better understanding of the most common weaknesses that can affect software security. With that in mind, there are many resources available online to help organizations learn about

Resources available to help organizations protect systems in

Resource	Focus
DoD Information Assurance Certification and Accreditation Process (DIACAP)	The DIACAP defines the minimum standa accredited by the DoD and authorized to application-level security controls, but it i activities, general tasks, and a managem
Defense Information Systems Agency (DISA)	The DISA provides a security technical im development that offer more granular info bility assessment techniques. The checklist is the same one used by DoD auditors.
U.S. Department of Homeland Security (DHS)	The DHS offers information on security best practices and tools for application- and soft part of its "Build Security In" initiative.
The Common Weaknesses Enumeration project, a community-based program sponsored by the MITRE Corporation, an IBM Business Partner	The MITRE Corporation maintains the online common vulnerabilities and exposures (CVE enumeration (CWE) knowledge bases about currently known vulnerabilities and types of knowledge base focuses on packaged software and deals with patches and known vul knowledge base focuses on code vulnerabilities.
The Open Web Application Security Project (OWASP)	One of the best sources for information on web application security issues, the OWASP 10 list of the most dangerous and most commonly found and commonly exploited vulne how to identify, fix and avoid them.
Cigital Building Security In Maturity Model (BSIMM)	Created by Cigital, an IBM Business Partner, the BSIMM is designed to help organizatio and plan a software security initiative. The focus is on making applications more secure, process and at later stages in the software life cycle.
IBM X-Force™ research and development team	A global cyberthreat and risk analysis team that monitors traffic and attacks around the IBM X-Force team is an excellent resource for trend analysis and answers to questions a attacks are most common, where they are coming from and what organizations can do the risks.
IBM Institute for Advanced Security (IAS)	This companywide cybersecurity initiative applies IBM research, services, software and t help governments and other clients improve the security and resiliency of their IT and bu

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 E be tested. Thi information ar and vulnerabi against the m

Creating a se plan includes

5 For more inforr
6 For more inforr

IBM

Security in Development: The IBM Secure Engineering Framework

Redguides for Business Leaders

Danny Allan
Tim Hahn
Andras Szakal
Jim Whitmore
Axel Buecker

- Investigating common development processes and the IBM Integrated Product Development process
- Emphasizing security awareness and requirements in the software development process
- Discussing test and vulnerability assessments

Redbooks

Software Engineering Institute

Making the Business Case for Software Assurance

Nancy R. Mead
Julia H. Allen
W. Arthur Conklin
Antonio Drommi
John Harrison
Jeff Ingalsbe
James Rainey
Dan Shoemaker

April 2009

SPECIAL REPORT
CMU/SEI-2009-SR-001

CERT Program

http://www.sei.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) 678-915-3718
jwang@spsu.edu

ABSTRACT

In order to reach the goals of the Information Security Automation Program (ISAP) [1], we propose an ontological approach to capturing 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 inference rules, knowledge representation, 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 successful.

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 remediation as well as automation of technical compliance activities. Its low-level objectives include enabling standards based communication of vulnerability data, customizing and managing configuration baselines for various IT products, assessing information systems and reporting compliance status, using standard metrics to weight and aggregate potential vulnerability impact, and remediating identified vulnerabilities [1]. Secure computer systems ensure that confidentiality, integrity, and availability are maintained for users, data, and other information assets. 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 stake holders due to the language ambiguity. On the other hand, the standardization, 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 relationship. Ontology represents knowledge in a formal and structured form. Therefore, ontology provides a better tool for communication, reusability, and organization of knowledge. Ontology is a knowledge representation (KR) system based on Description Logics (DLs) [6], which is an umbrella name for a family of KR formalisms representing knowledge in various domains. The DL formalism specifies a knowledge domain as the "world" by first defining the relevant concepts of the domain, and then it 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 logic inference based on well-defined data and knowledge bases. Ontology captures the relationships between collected data and use 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, properties, axioms and constraints.

A vulnerability is a security flaw, which arises from computer system 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 MITRE 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 (ontology 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. Vendors and users 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

CSIIRW '09, April 13-15, Oak Ridge, Tennessee, USA

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.

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

stablish a security plan across your

Industry Uptake

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.

SAFECode
Software Assurance Forum for Excellence in Code
Driving Security and Integrity

Fundamental Practices for Secure Software Development
2ND EDITION

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

February 8, 2011

CWE References

Much of CWE focuses on implementation issues, and Threat Modeling is a design-time event. There are, however, a number of CWEs that are applicable to the threat modeling process, including:

- CWE-287: Improper authentication is an example of weakness that could be exploited by a Spoofing threat
- CWE-264: Permissions, Privileges, and Access Controls is a parent weakness of many Tampering, Repudiation and Elevation of Privilege threats
- CWE-311: Missing Encryption of Sensitive Data is an example of an Information Disclosure threat
- CWE-400: (uncontrolled resource consumption) is one example of an unmitigated Denial of Service threat

An example of a portion of a test plan derived from a Threat Model could be:

Threat Identified	Design Element(s)	Mitigation	Verification
Session Hijacking	GUI	Ensure random session identifiers of appropriate length	Collect session identifiers over a number of sessions and examine distribution and length
Tampering with data in transit	Process A on server to Process B on client	Use SSL to ensure that data isn't modified in transit	Assert that communication cannot be established without the use of SSL

CWE

OWASP
The Open Web Application Security Project

Code Review Introduction

««Code Review Guide History«« | Main (Table of Contents) | »»Preparation»»

Contents [hide]

Introduction

Code review is probably the single-most effective technique for identifying security flaws. When used together with automated tools and manual penetration testing, code review can significantly increase the cost effectiveness of an application security verification effort.

This guide does not prescribe a process for performing a security code review. Rather, this guide focuses on the mechanics of reviewing code for certain vulnerabilities, and provides limited guidance on how the effort should be structured and executed. OWASP intends to develop a more detailed process in a future version of this guide.

Manual security code review provides insight into the "real risk" associated with insecure code. This is the single most important value from a manual approach. A human reviewer can understand the context for certain coding practices, and make a serious risk estimate that accounts for both the likelihood of attack and the business impact of a breach.

Why Does Code Have Vulnerabilities?

MITRE has catalogued almost 700 different kinds of software weaknesses in their CWE project. These 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.

One goal for this project is to help software buyers gain visibility into the security of software and start to effect change in the software market.

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:

"We never get hacked (that I know of), we don't need security"

The Web Application Security Consortium / Threat Classification Taxonomy Cross Reference View

http://projects.webappsec.org/w/page/13246975/Threat-Classification-Taxonomy-Cross-Reference-View

Google

The Web Application Security Consortium

log in help

Wiki Pages & Files

Search this workspace

VIEW

Threat Classification Taxonomy Cross Reference View

last edited by Robert Auger 10 months, 3 weeks ago

Page history

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 RC1 (original mapping with OWASP Top Ten from Jeremiah Grossman & Bill Corry) and SANS/CWE and OWASP Top Ten 2007 and 2004 (original mapping from Dan Cornell, Denim Group)

WASC ID	Name	CWE ID	CAPEC ID	SANS/CWE Top 25 2009	OWASP Top Ten 2010	OWASP Top Ten 2007	OWASP Top Ten 2004
WASC-01	Insufficient Authentication	287		642	A3 - Broken Authentication and Session Management, A4 - Insecure Direct Object References	A7 - Broken Authentication and Session Management, A4 - Insecure Direct Object Reference	A3 - Broken Authentication and Session management, A2 - Broken Access Control
WASC-02	Insufficient Authorization	284		285	A4 - Insecure Direct Object References, A7 - Failure to Restrict URL Access	A10 - Failure to Restrict URL Access, A4 - Insecure Direct Object Reference	A2 - Broken Access Control
WASC-03	Integer Overflows	190	128	682			
WASC-04	Insufficient Transport Layer Protection	311 523		319	A10 - Insufficient Transport Layer Protection	A9 - Insecure Communications	
WASC-05	Remote File Inclusion	98	193 253	426		A3 - Malicious File Execution	
WASC-06	Format String	134	67				
WASC-07	Buffer Overflow	119 120	10 100	119			A5 - Buffer Overflows
WASC-08	Cross-site Scripting	79	18 19 63	79	A2 - Cross-Site Scripting	A1 - Cross Site Scripting (XSS)	A4 - Cross Site Scripting (XSS)
WASC-09	Cross-site Request Forgery	352	62	352	A5 - Cross-Site Request Forgery	A5 - Cross Site Request Forgery (CSRF)	
WASC-10	Denial of Service	400	119	404	A7 - Failure to Restrict	A10 - Failure to	A9 - Denial of

Tags: Threat Classification

Check for plagiarism

SideBar

WASC Projects

- Distributed Open Proxy Honeypots
- Script Mapping
- The Web Security Glossary
- Web Application Firewall Evaluation Criteria
- Web Application Security Scanner Evaluation Criteria
- Web Application Security Statistics
- Web Hacking Incidents Database
- WASC Threat Classification

WASC Project Leaders

- Robert Auger
- Ryan Barnett
- Romain Gaucher
- Sergey Gordeychik
- Ofer Shezaf
- Brian Shura

WASC Main Website

- http://www.webappsec.org/

WASC Mailing Lists

- http://lists.webappsec.org/

WASC on Twitter

- http://twitter.com/wascupdates

Join us on Linkedin!

- http://www.linkedin.com/groups?gid=83336

Recent Activity

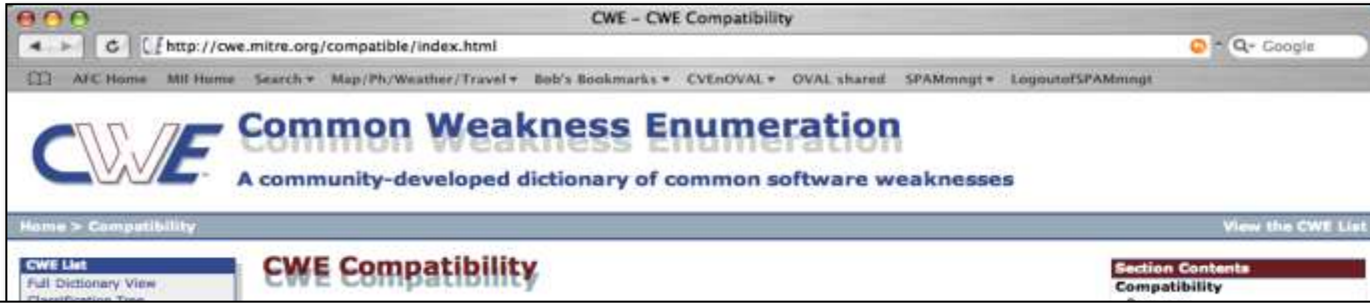
Insufficient Data Protection Working

edited by Robert Auger

Done

CWE Compatibility & Effectiveness Program

(launched Feb 2007)



Organizations Participating

cwe.mitre.org/compatible/

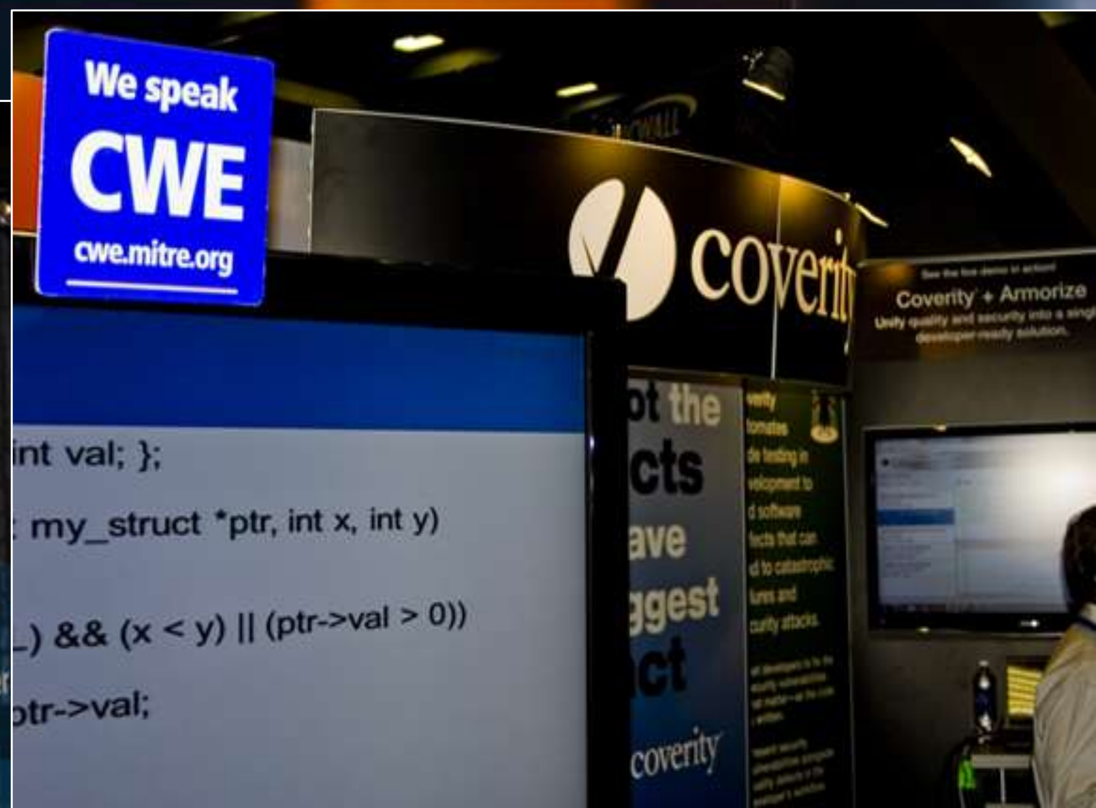
All organizations participating in the CWE Compatibility and Effectiveness Program are listed below, including those with CWE-Compatible Products and Services and those with Declarations to Be CWE-Compatible.

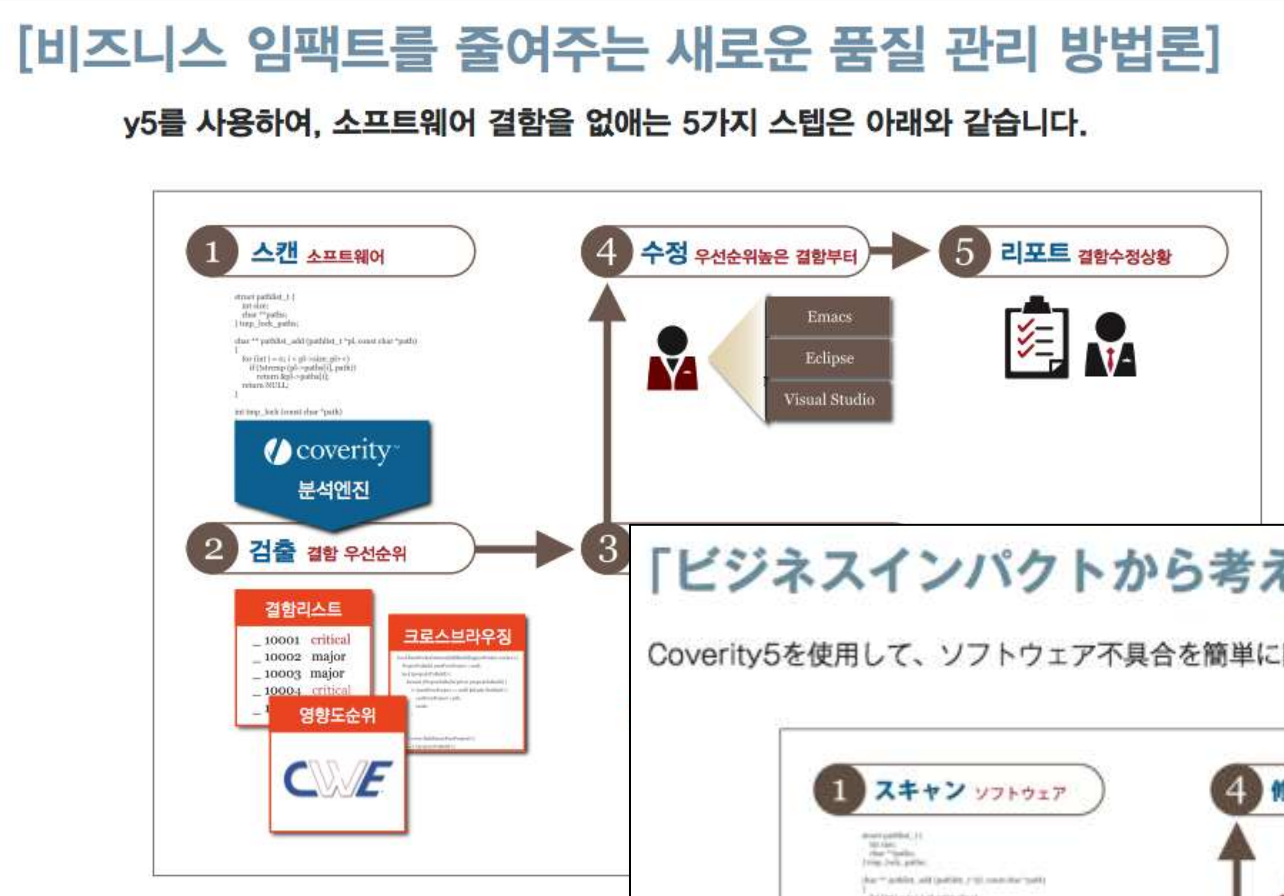
TOTALS
Organizations Participating: **31**
Products & Services: **53**

Products are listed alphabetically by organization name:

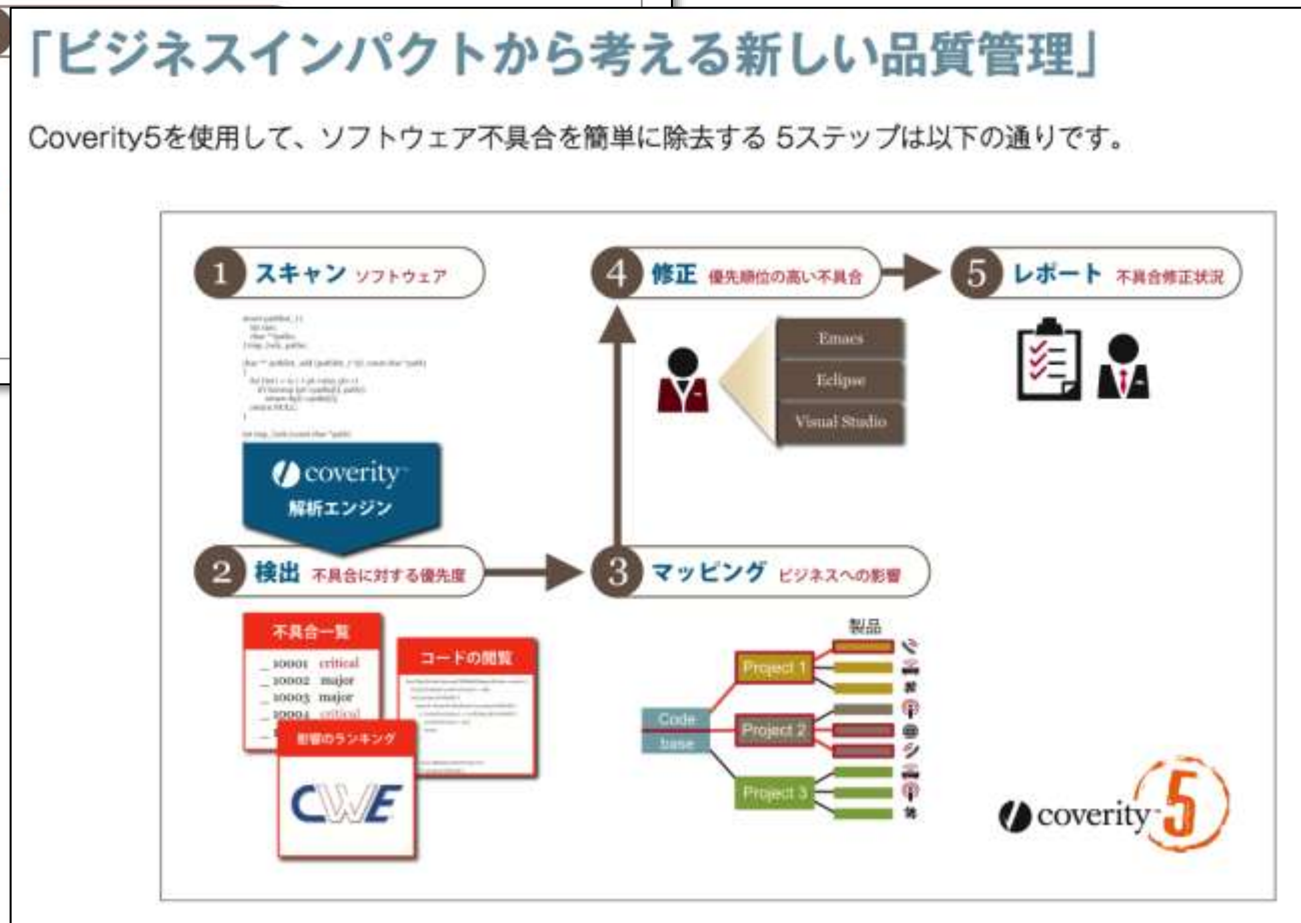

armorize
The Web Malware Experts
We speak
CWE
cwe.mitre.org


We speak
CWE
cwe.mitre.org
We speak
CVE
cve.mitre.org
GLOBAL INFORMATION ASSURANCE CERTIFICATION
GIAC
SANS


We speak
CWE
cwe.mitre.org
coverity
Coverity + Armorize
int val; };
my_struct *ptr, int x, int y)
) && (x < y) || (ptr->val > 0))
ptr->val;
coverity



Korean



Japanese

CWE Coverage – Implemented…

Coverity® Data Sheet

coverity®

Coverity Coverage for Common Weakness Enumeration (CWE): Java

CWE ID	Coverity Static Analysis Checker
171	BAD_EQ
252	CHECKED_RETURN
366	GUARDED_BY_VIOLATION
	INDIRECT_GUARDED_BY_VIOLATION
	NON_STATIC_GUARDING_STATIC
	VOLATILE_ATOMICITY
382	DC.CODING_STYLE
	BAD_OVERRIDE
	DC.EXPLICIT_DEPRECATION
	DC.GC
	MUTABLE_COMPARISON
398	MUTABLE_HASHCODE

Coverity® Data Sheet

coverity®

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

CWE ID	Coverity Static Analysis Checker	Checker Description	Type of Security Risk
	TAINTED_SCALAR	Use of untrusted scalar value	Alter control flow; Arbitrary control of a resource; Arbitrary code execution

CENZIC www.cenzic.com | (866) 4-CENZIC (866-423-6942)

Cenzic Product Suite is CWE Compatible

Cenzic Hailstorm Enterprise ARC, Cenzic Hailstorm Professional and Cenzic ClickToSecure are compatible with the CWE standard or Common Weakness Enumeration as maintained by Mitre Corporation. Web security assessment results from the Hailstorm product suite are mapped to the relevant CWE ID's providing users with additional information to classify and describe common weaknesses found in Web applications.

For additional details on CWE, please visit: http://cwe.mitre.org/index.html

The following is a mapping between Cenzic's SmartAttacks and CWE ID's:

	Cenzic SmartAttack Name	CWE ID/s
1	Application Exception	**CWE-388:** Error Handling
2	Application Exception (WS)	**CWE-388:** Error Handling
3	Application Path Disclosure	**CWE-200:** Information Leak (rough match)
4	Authentication Bypass	**CWE-89:** Failure to Sanitize Data into SQL Queries (aka 'SQL Injection') (rough match)
5	Authorization Boundary	**CWE-285:** Missing or Inconsistent Access Control, CWE-425: Direct Request ('Forced Browsing')
6	Blind SQL Injection	**CWE-89:** Failure to Sanitize Data into SQL Queries (aka 'SQL Injection')
7	Blind SQL Injection (WS)	**CWE-89:** Failure to Sanitize Data into SQL Queries (aka 'SQL Injection')
8	Browse HTTP from HTTPS List	**CWE-200:** Information Leak
9	Brute Force Login	**CWE-521:** Weak Password Requirements
10	Buffer Overflow	**CWE-120:** Unbounded Transfer ('Classic Buffer Overflow')
11	Buffer Overflow (WS)	**CWE-120:** Unbounded Transfer ('Classic Buffer Overflow')
12	Check Basic Auth over HTTP	**CWE-200:** Information Leak
13	Check HTTP Methods	**CWE-650:** Trusting HTTP Permission Methods on the Server Side

Cenzic CWE Brochure | October 2009 1

CWE IDs mapped to Klocwork Java issue types - current http://www.klocwork.com/products/documentation/curren...

CWE IDs mapped to Klocwork Java issue types

From current

CWE IDs mapped to Klocwork Java issue types

See also Detected Java Issues.

CWE IDs mapped to Klocwork C and C++ issue types/ja - ... http://www.klocwork.com/products/documentation/curren...

CWE IDs mapped to Klocwork C and C++ issue types/ja

From current

< CWE IDs mapped to Klocwork C and C++ issue types
CWE IDs mapped to Klocwork C and C++ issue types/ja

その他の情報 Detected C and C++ Issues.

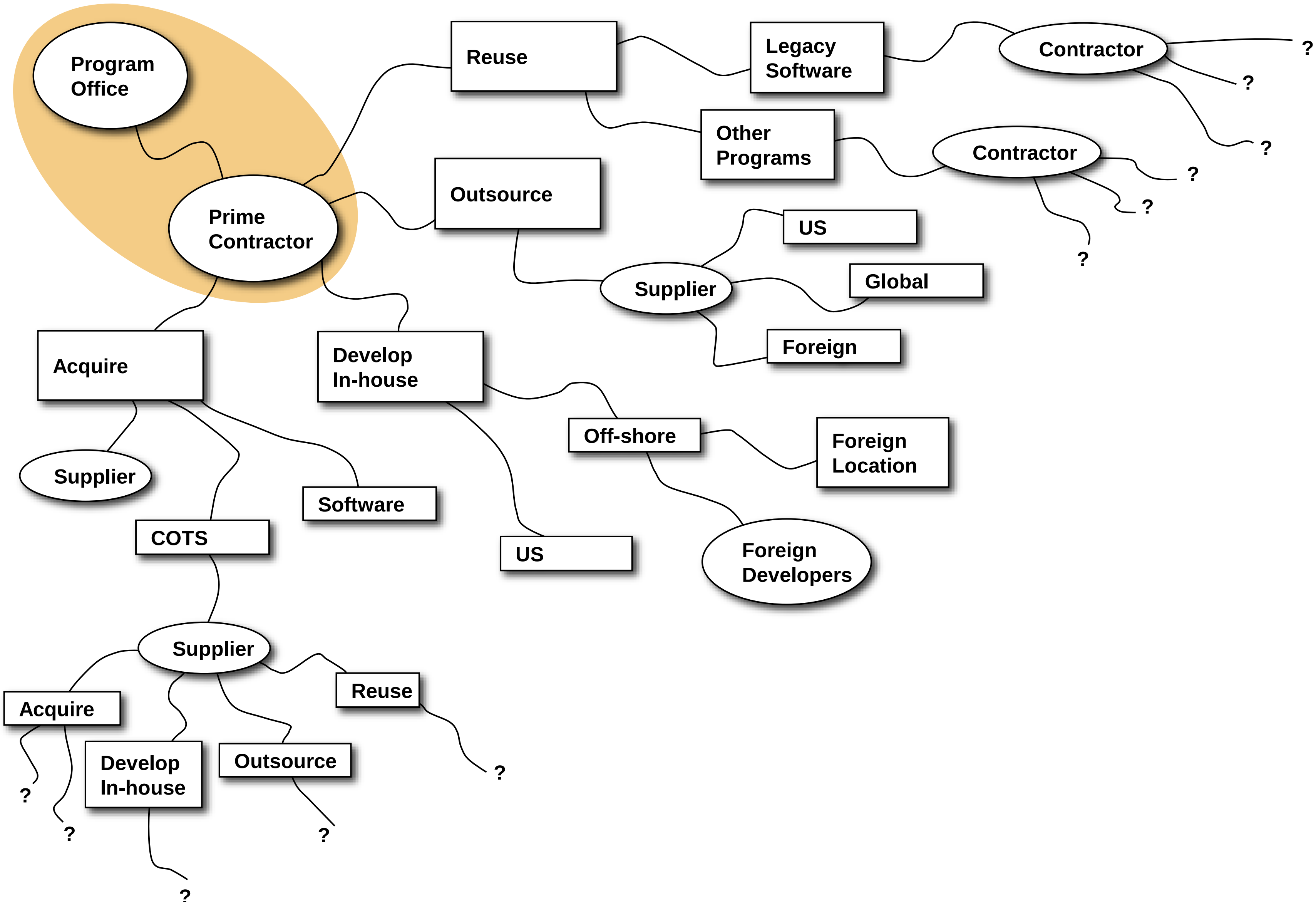
CWE ID	説明
20 (http://cwe.mitre.org/data/definitions/20.html)	ABV.TAINTED 未検証入力によるバッファ オーバーフロー SV.TAINTED.GENERIC 未検証文字列データの使用 SV.TAINTED.ALLOC_SIZE メモリ割り当てにおける未検証の整数の使用 SV.TAINTED.CALL.INDEX_ACCESS 関数呼び出しにおける未検証整数の配列インデックスとしての使用
22 (http://cwe.mitre.org/data/definitions/22.html)	SV.CUDS.MISSING_ABSOLUTE_PATH ファイルのロードでの絶対パスの不使用
73 (http://cwe.mitre.org/data/definitions/73.html)	SV.CUDS.MISSING_ABSOLUTE_PATH ファイルのロードでの絶対パスの不使用
74 (http://cwe.mitre.org/data/definitions/74.html)	SV.TAINTED.INJECTION コマンド インジェクション
77 (http://cwe.mitre.org/data/definitions/77.html)	SV.CODE_INJECTION.SHELL_EXEC シェル実行へのコマンド インジェクション
78 (http://cwe.mitre.org/data/definitions/78.html)	NNTS.TAINTED 未検証ユーザ入力が原因のバッファ オーバーフロー - 非 NULL 終端文字列 SV.TAINTED.INJECTION コマンド インジェクション
88 (http://cwe.mitre.org	SV.TAINTED.INJECTION コマンド インジェクション NNTS.TAINTED 未検証ユーザ入力が原因のバッファ オーバーフロー

1 of 7 2/26/11 10:34 AM

2/26/11 10:35 AM

The Software Supply Chain

*



* "Scope of Supplier Expansion and Foreign Involvement" graphic in DACS www.softwaretechnews.com Secure Software Engineering, July 2005 article "Software Development Security: A Risk Management Perspective" synopsis of May 2004 GAO-04-678 report "Defense Acquisition: Knowledge of Software Suppliers Needed to Manage Risks"

What Is an Assurance Case?

History of Assurance Cases

- Originally Only Safety Cases
  - **Aerospace**
  - **Railways, automated passenger**
  - **Nuclear power**
  - **Off-shore oil**
  - **Defense**
- Security Cases
  - **Use compliance rules more than an assurance case**
- Cases for Business Critical Systems

Definition of Safety Case

- From Adelard's ASCE manual:

"A documented body of evidence that provides a convincing and valid argument that a system is adequately safe for a given application in a given environment."

Definition of Assurance Case

- Generalizing that definition

  A documented body of evidence that provides a convincing and valid argument that a specified set of critical claims regarding a system’s properties are adequately justified for a given application in a given environment.

Structured Assurance Cases

- Structure is required to make the creation, sharing, analysis, maintenance and automation of such an assurance case practical

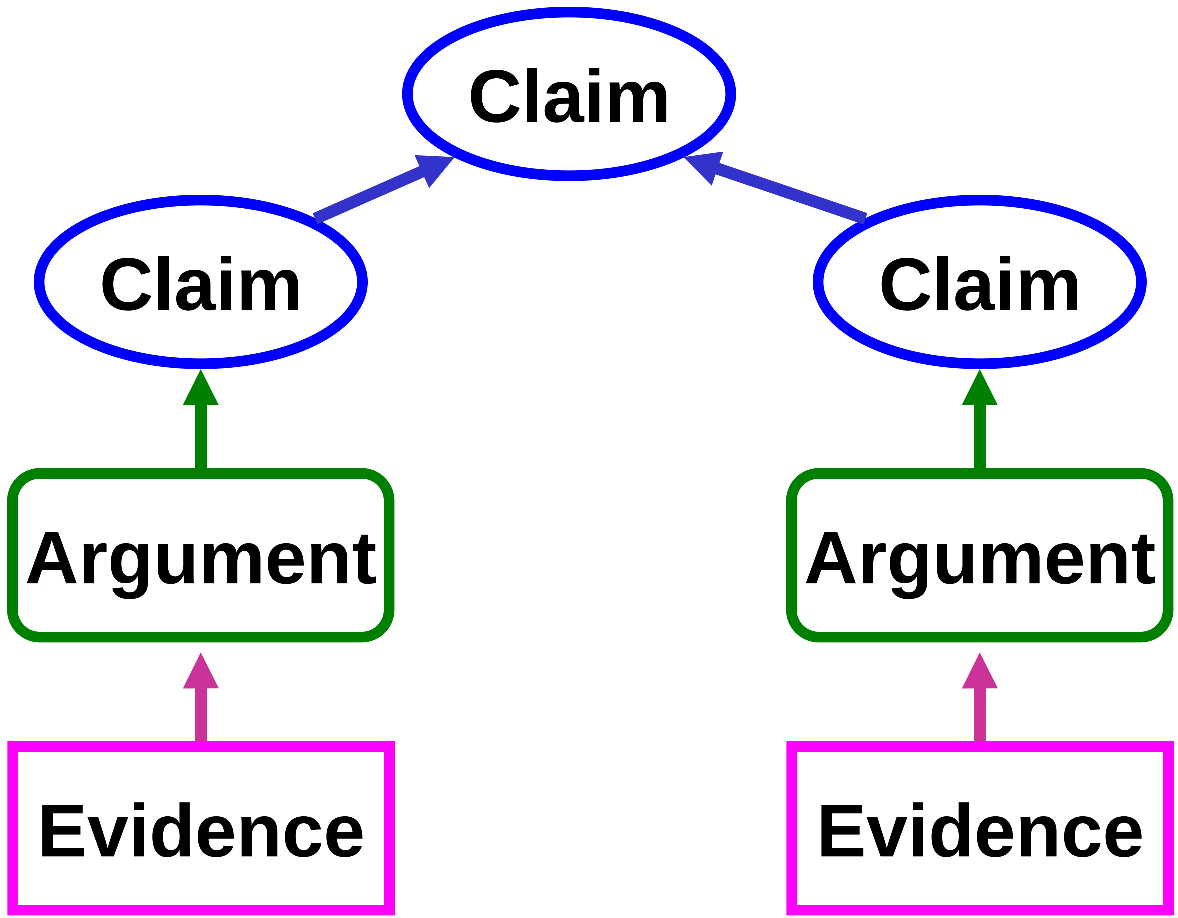
- Structured Assurance Cases are composed of structured sets of Claims, Arguments and Evidence
  - **A Claim is a proposition to be assured about the system of concern**
  - **An Argument is a reasoning of why a claim is true**
  - **Evidence is either a fact, a datum, an object, a claim or [recursively] an assurance case which supports an Argument against a Claim**

Extremely Simplified Overview of Structured Assurance Case Content

Claim =
assertion to be proven

Argument =
reasoning supporting a claim

Evidence =
data supporting an Argument



Need for Standards

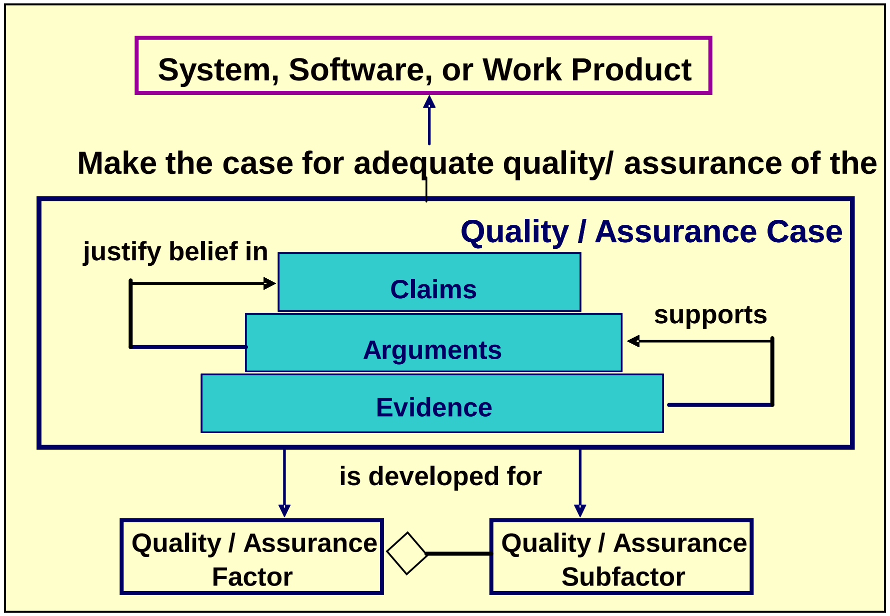
- While several different notations exist for safety cases and generalized assurance cases no widely accepted standard currently exists for specifying structured assurance cases within a systems & software assurance domain
- Standards are needed before structured assurance cases can be widely leveraged or made practical through automated tooling
- Coordinated efforts are currently underway in the International Standards Organization (ISO) and the Object Management Group (OMG) to develop these needed standards
  - **ISO 15026 Part 2 (currently published) is a very simple high-level standard outlining the context and basic requirements for structured assurance cases**
  - **The OMG SACM (under development) and supporting OMG standards are targeted at providing at automatable level of detail for structured assurance case specification**

ISO/IEC 15026: A Four-Part Standard

- Planned parts:
  - **15026-1: Concepts and vocabulary (initially a TR2 and then revised to be an IS)**
  - **15026-2: Assurance case (including planning for the assurance case itself)**
  - **15026-3: System integrity levels (a revision of the 1998 standard)**
  - **15026-4: Assurance in the life cycle (including project planning for assurance considerations)**
- Possible additional parts as demand requires and resources permit, e.g.
  - **Assurance analyses and techniques**
  - **Guidance documents**

ISO/IEC/IEEE 15026 Assurance Case

- Set of structured assurance claims, supported by evidence and reasoning (arguments), that demonstrates how assurance needs have been satisfied.
  - Shows compliance with assurance objectives
  - Provides an argument for the safety and security of the product or service.
  - Built, collected, and maintained throughout the life cycle
  - Derived from multiple sources



- Sub-parts
  - A high level summary
  - Justification that product or service is acceptably safe, secure, or dependable
  - Rationale for claiming a specified level of safety and security
  - Conformance with relevant standards & regulatory requirements
  - The configuration baseline
  - Identified hazards and threats and residual risk of each hazard / threat
  - Operational & support assumptions

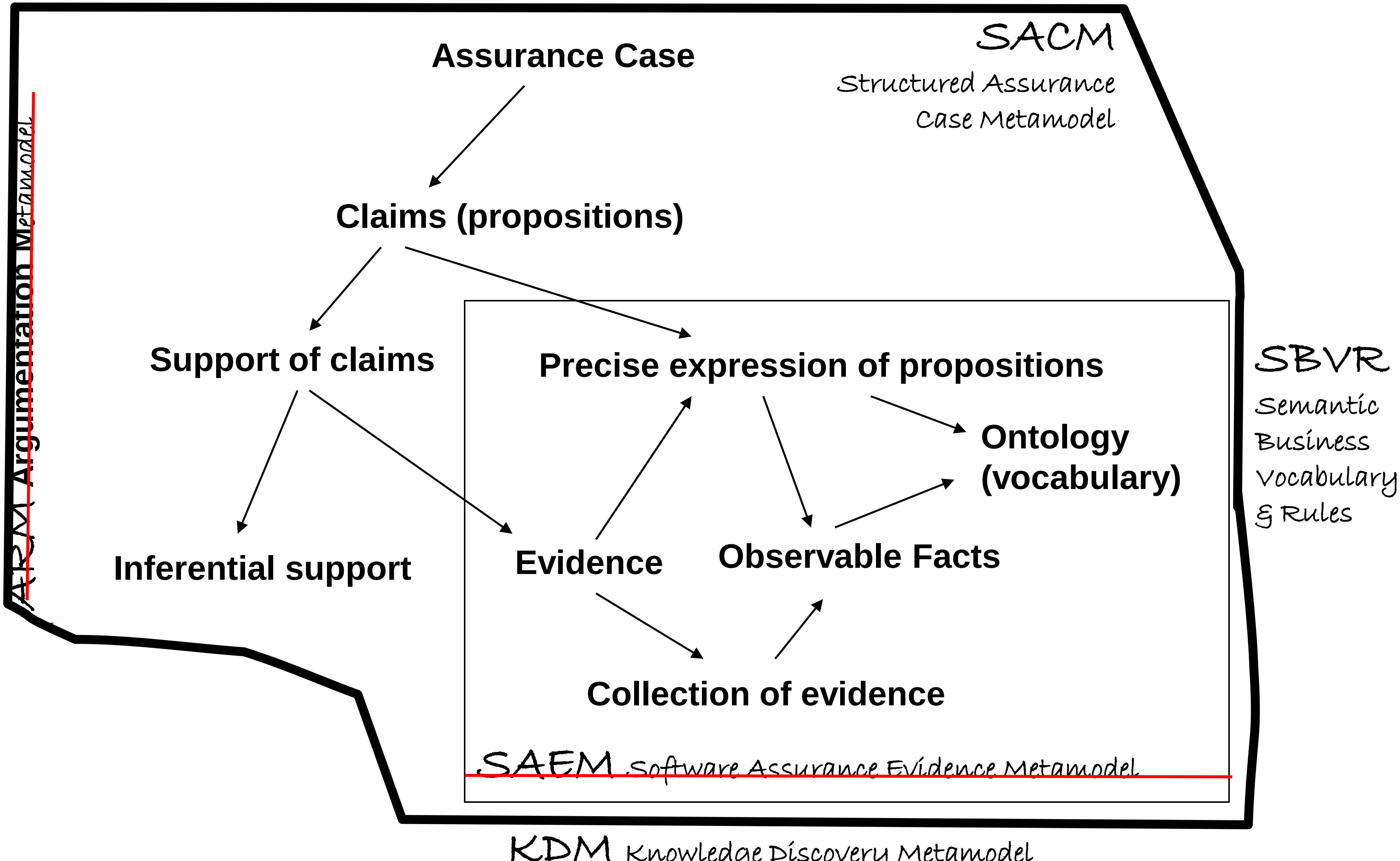
Attributes

- ❑ Clear
- ❑ Consistent
- ❑ Complete
- ❑ Comprehensible
- ❑ Defensible
- ❑ Bounded
- ❑ Addresses all life cycle stages

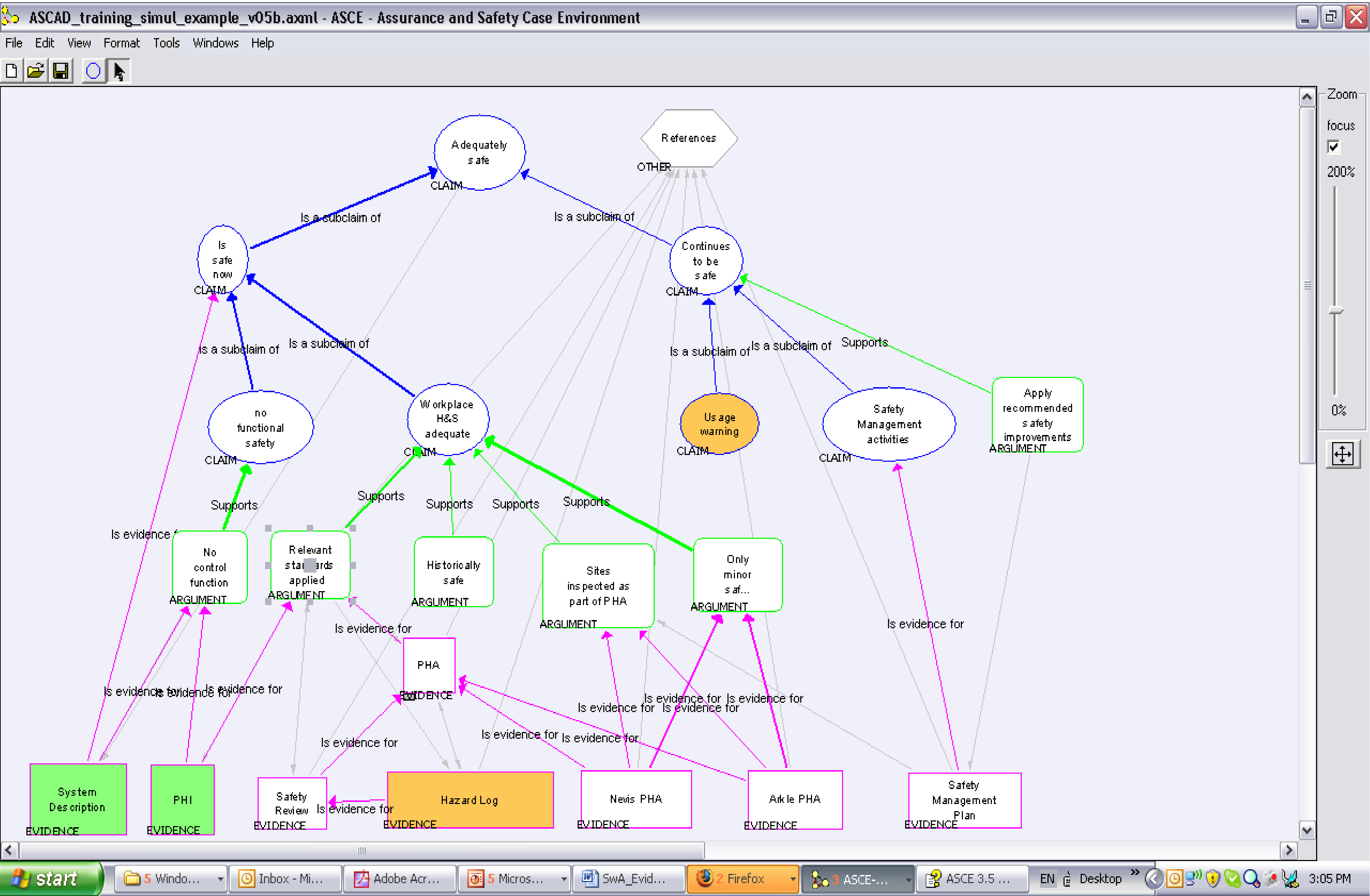
Structured Assurance Case Efforts at the OMG

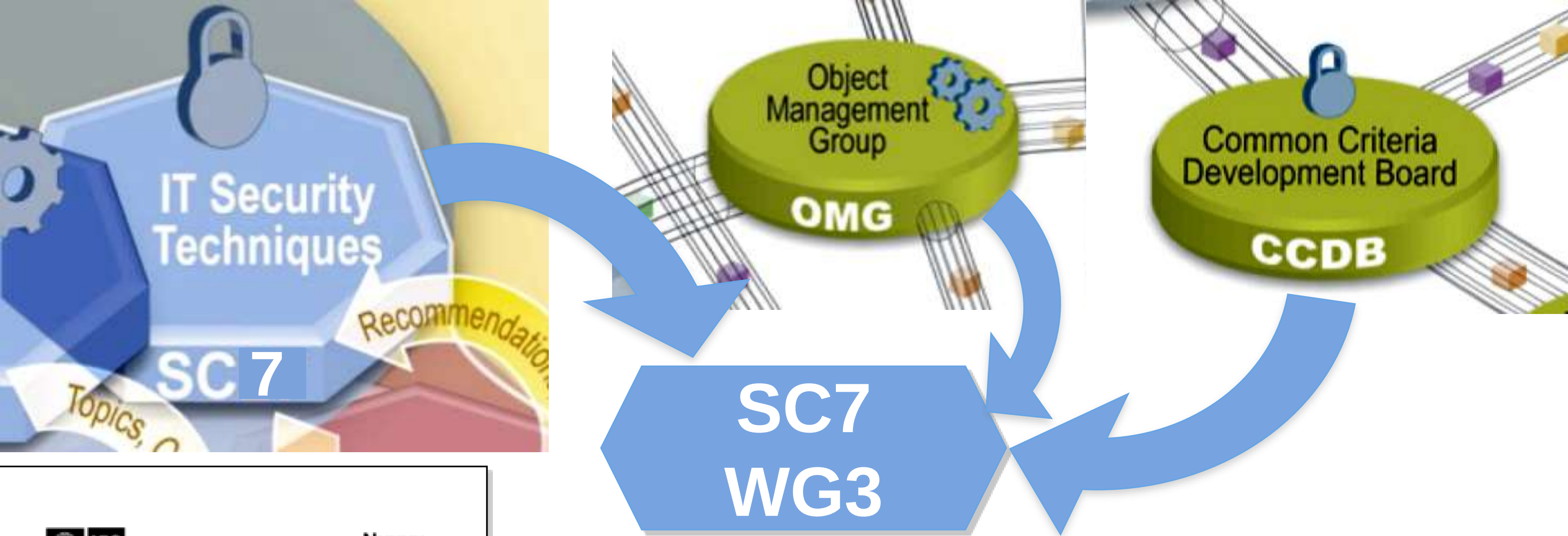
- There are efforts underway within the Object Management Group (OMG) to leverage existing standards and develop new standards for specifying ISO 15026 structured assurance cases in such a way that they will fully support automation
  - **Currently working to integrate two draft standards (the Argumentation Metamodel (ARM) and the Software Assurance Evidence Metamodel (SAEM)) into a single standard (Structured Assurance Case Metamodel (SACM)) for structured assurance case specification**
  - **SACM will also likely leverage the existing OMG Knowledge Discovery Metamodel (KDM) and Semantic Business Vocabulary & Rules (SBVR) standards**

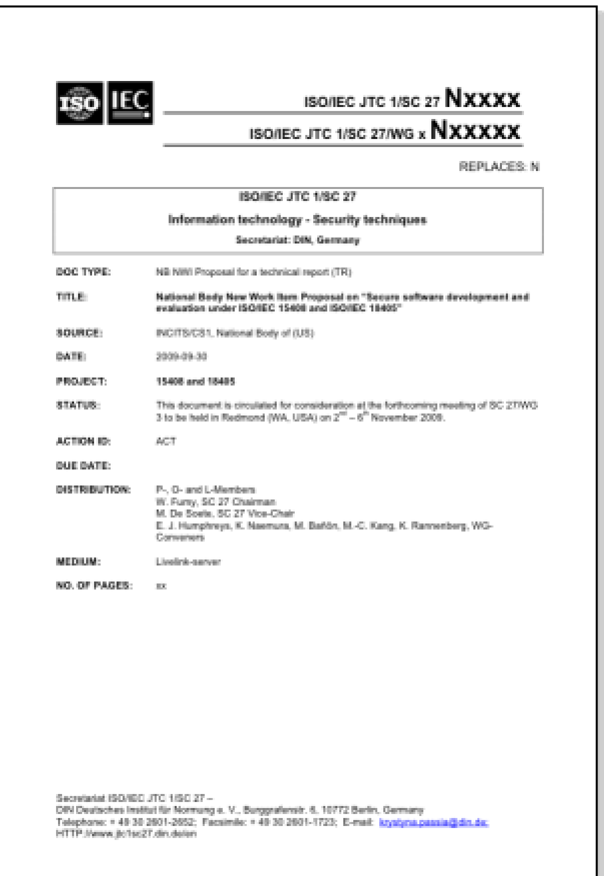
Object Management Group (OMG)
Systems Assurance Task Force
Claims-Evidence-Arguments Overview



Structured Safety Assurance tools are commercially available






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

REPLACES: N

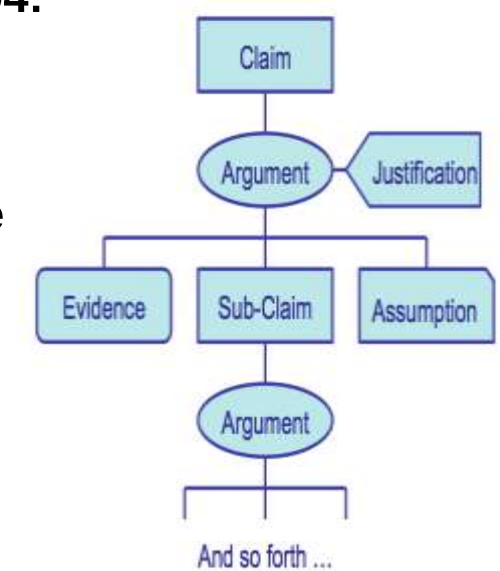
ISO/IEC JTC 1/SC 27
Information technology - Security techniques
Secretariat: DIN, Germany

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

Common Criteria v4 CCDB
- TOE to leverage CAPEC & CWE
- ISO/IEC JTC 1/SC 7/WG 3, TR 20004: "Refining Software Vulnerability Analysis Under ISO/IEC 15408 and ISO/IEC 18045"
- Also investigating how to leverage ISO/IEC 15026 and OMG's Structured Assurance Case Metamodel (SACM)

NIAP (U.S.) Evaluation Scheme
- Above plus
- Also investigating how to leverage SCAP



OMG's Software Assurance Ecosystem: The Formal Framework

The value of formalization extends beyond software systems to include related software system process, people and documentation

Process Docs & Artifacts

Requirements/Design Docs & Artifacts

Reports
Risk Analysis, etc)

Process, People & Documentation Evaluation Environment

- Some point tools to assist evaluators but mainly manual work
- Claims in Formal SBVR vocabulary
- Evidence in Formal SBVR vocabulary
- Large scope requires large effort

Process, People, documentation Evidence

Formalized Specifications

Claims, Arguments and Evidence Repository

- Formalized in SBVR vocabulary
- Automated verification of claims against evidence
- Highly automated and sophisticated risk assessments using transitive inter-evidence point relationships

Software System / Architecture Evaluation

- Many integrated & highly automated tools to assist evaluators
- Claims and Evidence in Formal vocabulary
- Combination of tools and ISO/OMG standards
- Standardized SW System Representation In KDM
- Large scope capable (system of systems)
- Iterative extraction and analysis for rules

Software system Technical Evidence

Executable Specifications

Hardware Environment

Software System Artifacts

Protection Profiles

IA Controls

CWE-CAPEC

Contact Info

sbarnum@mitre.org

Making Security Measurable™

http://cwe.mitre.org

http://capec.mitre.org