Software & Supply Chain Assurance:

Enabling Enterprise Resilience through Software Assurance and Supply Chain Risk Management

Joe Jarzombek, PMP, CSSLP
Director for Software & Supply Chain Assurance
Stakeholder Engagement & Cyber Infrastructure Resilience
Cyber Security & Communications
joe.jarzombek@hq.dhs.gov

Homeland Security

Mitigating Cyber-Physical Risk Exposures Attributable to External Dependencies on ICT Supply Chain Components and Services

SOFTWARE AND SUPPLY CHAIN ASSURANCE

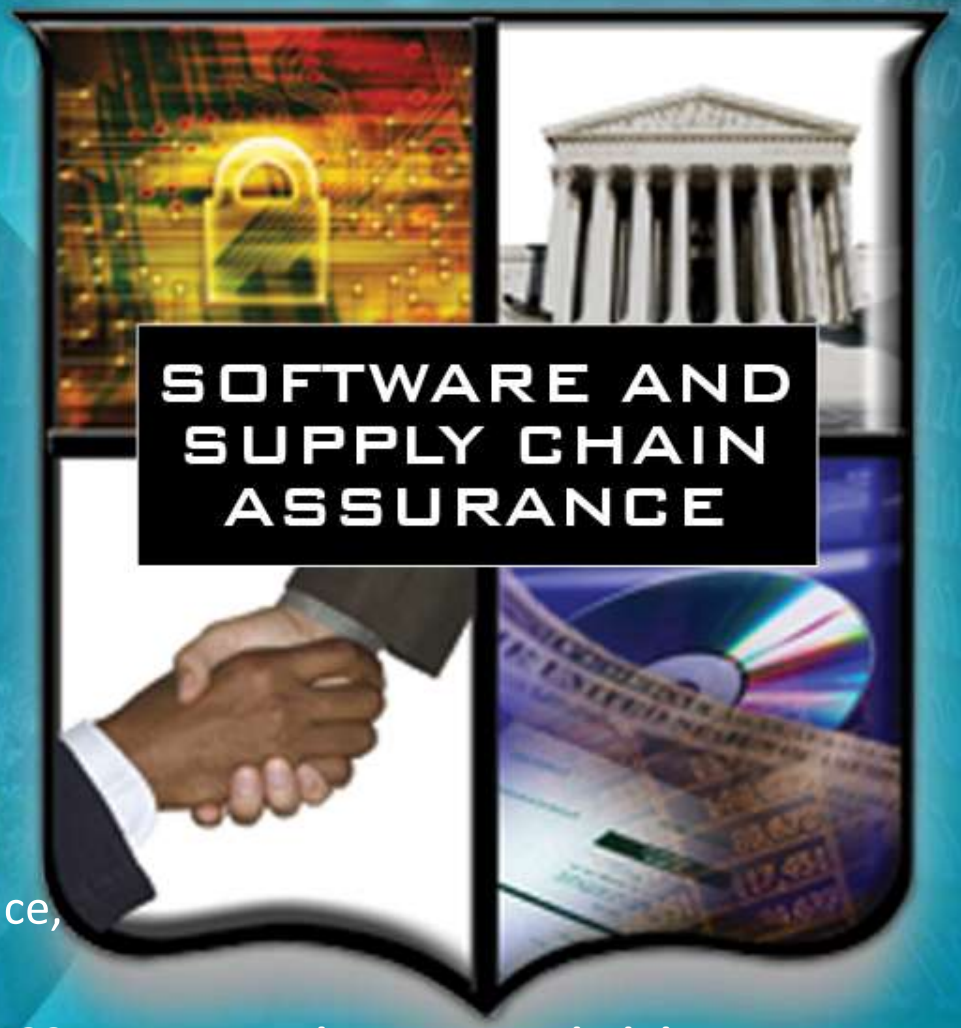
Homeland Security

National Defense

Commerce & Standards

General Services

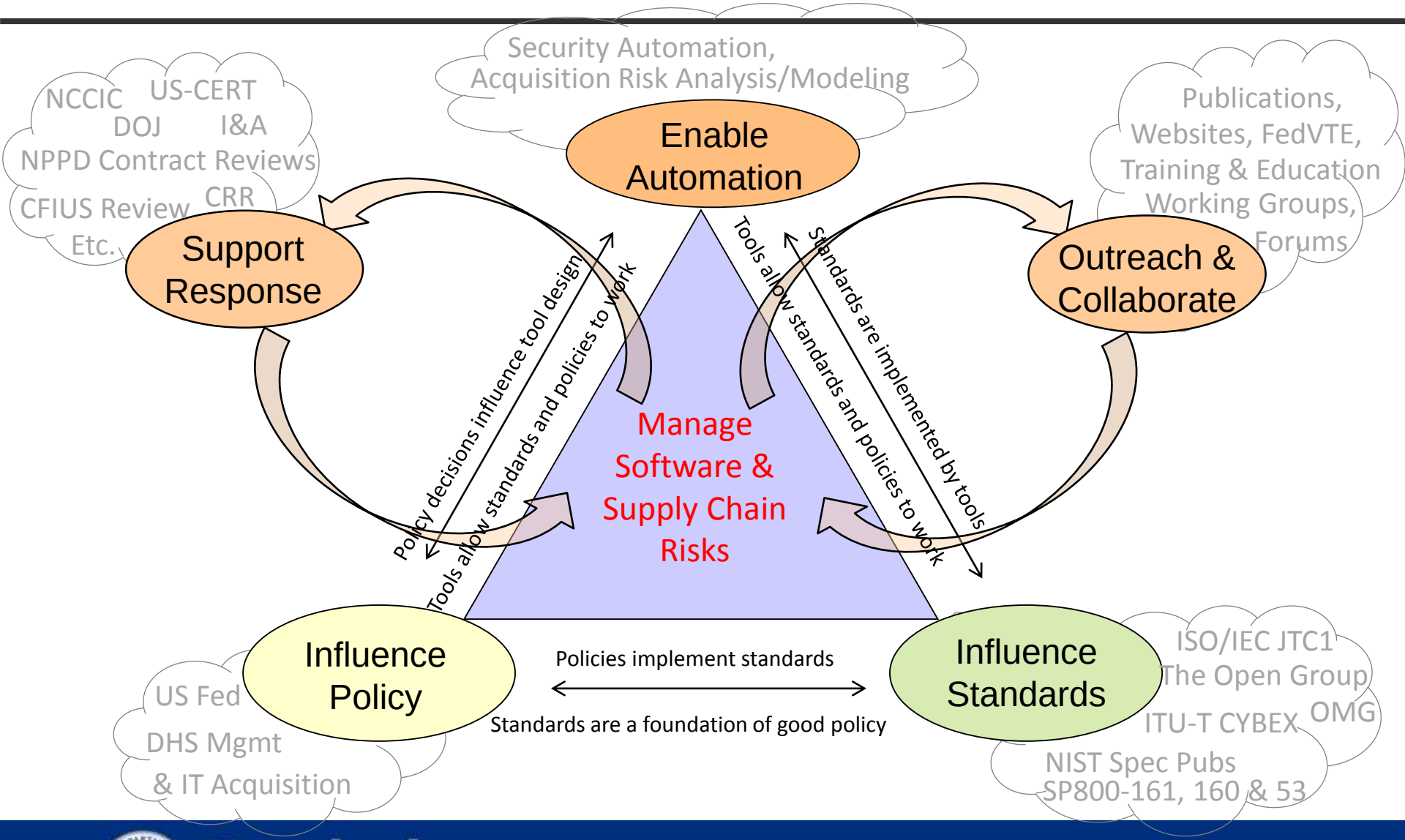
BUILDING SECURITY IN



Public-Private Collaboration Efforts for Security Automation, Software Assurance, and Supply Chain Risk Management

Next SSCA Forum 9-11 March 2015 at MITRE in McLean, Virginia

DHS Software & Supply Chain Assurance Strategy



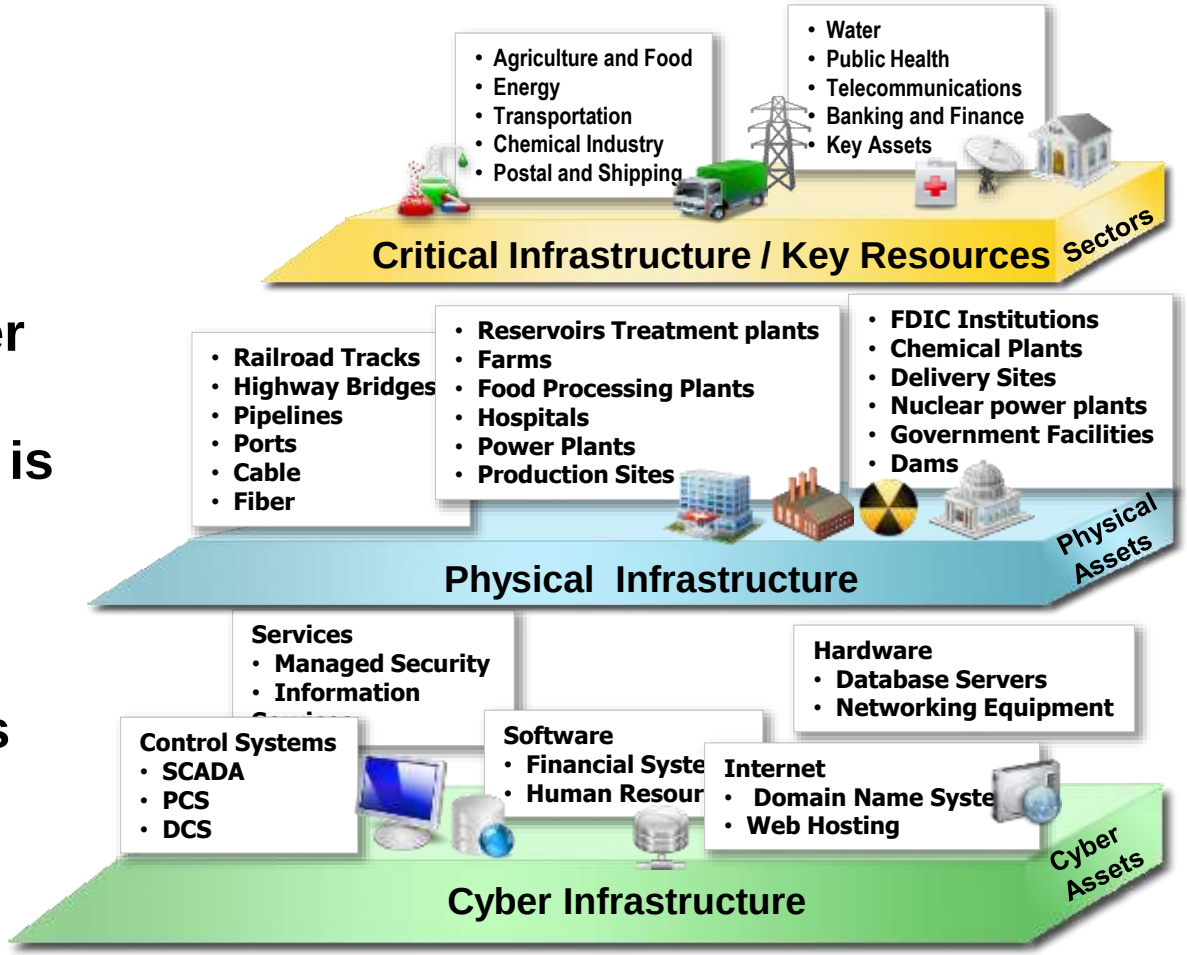
DHS Software & Supply Chain Assurance Outreach

- Co-sponsor SSCA Forum & WGs for government, academia, and industry to facilitate ongoing public-private collaboration.
- Co-sponsor issues of CROSSTALK to "spread the word"
  - Sep/Oct 2009 issue on "Resilient Software"
  - Mar/Apr 2010 issue on "System Assurance"
  - Sep/Oct 2010 issue on "Game Changing Tools & Practices"
  - Mar/Apr 2011 issue on "Rugged Software"
  - Sep/Oct 2011 issue on "Protecting against Predatory Practices"
  - Mar/Apr 2012 issue on "Securing a Mobile World"
  - Sep/Oct 2012 issue on "Resilient Cyber Ecosystem"
  - Mar/Apr 2013 issue on "Supply Chain Risk Management"
  - Sep/Oct 2013 issue on "Securing the Cloud"
  - Mar/Apr 2014 issue on "Mitigating Risks from Counterfeit & Tainted Products"
  - **Mar/Apr 2015 issue on "Test & Diagnostics"**
- Collaborate with standards organizations, consortiums, professional societies, education/training initiatives in promoting SwA
- Provide free SwA resources via "BuildSecurityIn" website to promote secure development methodologies (since Oct 05)
- Host SSCA Community Resources & Information Clearinghouse via https://buildsecurityin.us-cert.gov/SwA

Homeland Security

Gaining confidence in ICT/software-based cyber technologies

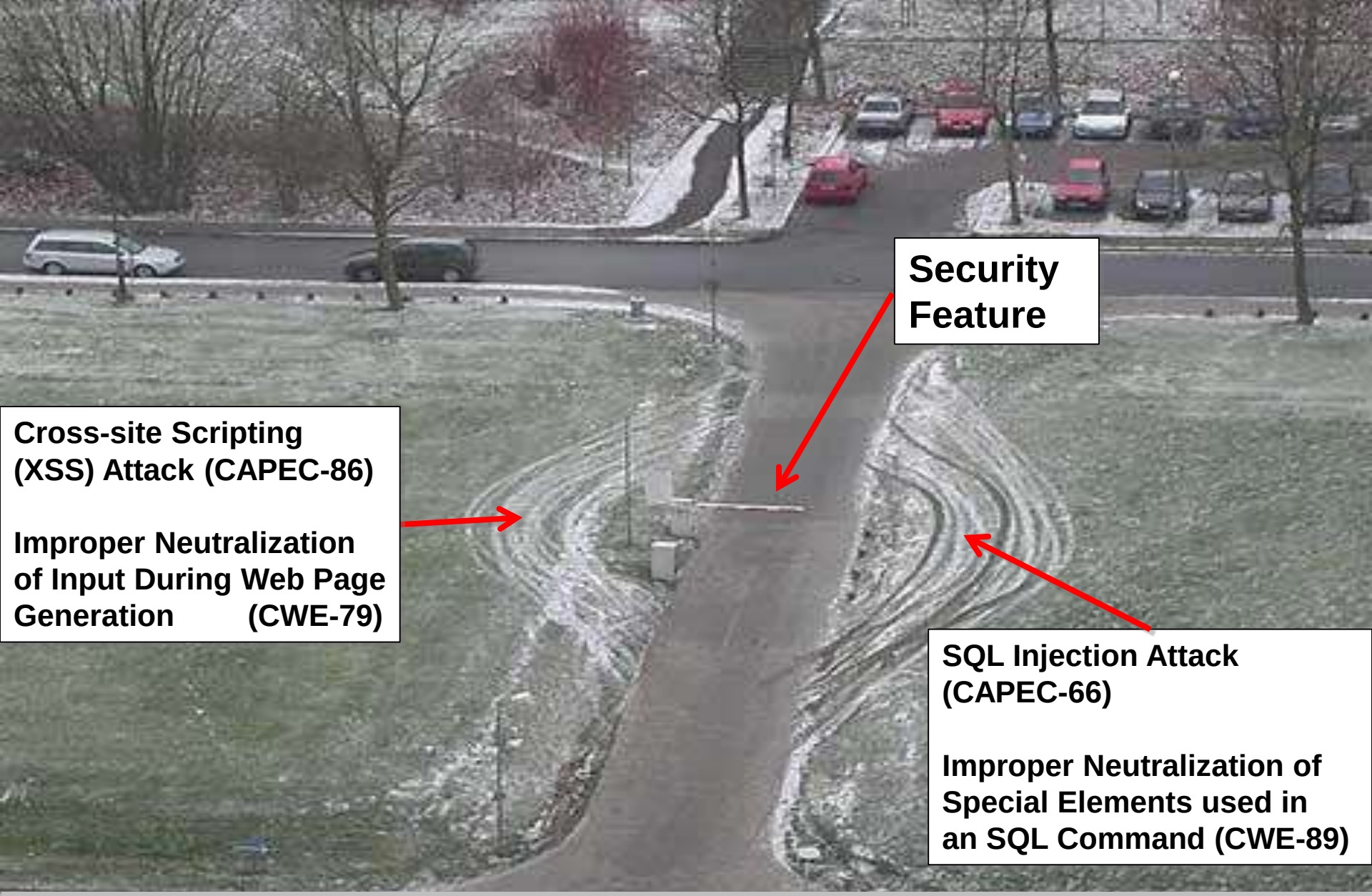
- Dependencies on technology are greater then ever
- Possibility of disruption is greater than ever because hardware/ software is vulnerable
- Loss of confidence alone can lead to stakeholder actions that disrupt critical business activities



Homeland Security

Interdependencies Between Physical & Cyber Capabilities –
Convergence of Safety, Security and Resilience Considerations

In an era riddled with asymmetric cyber attacks, claims about system reliability and safety must include provisions for built-in security of the enabling software



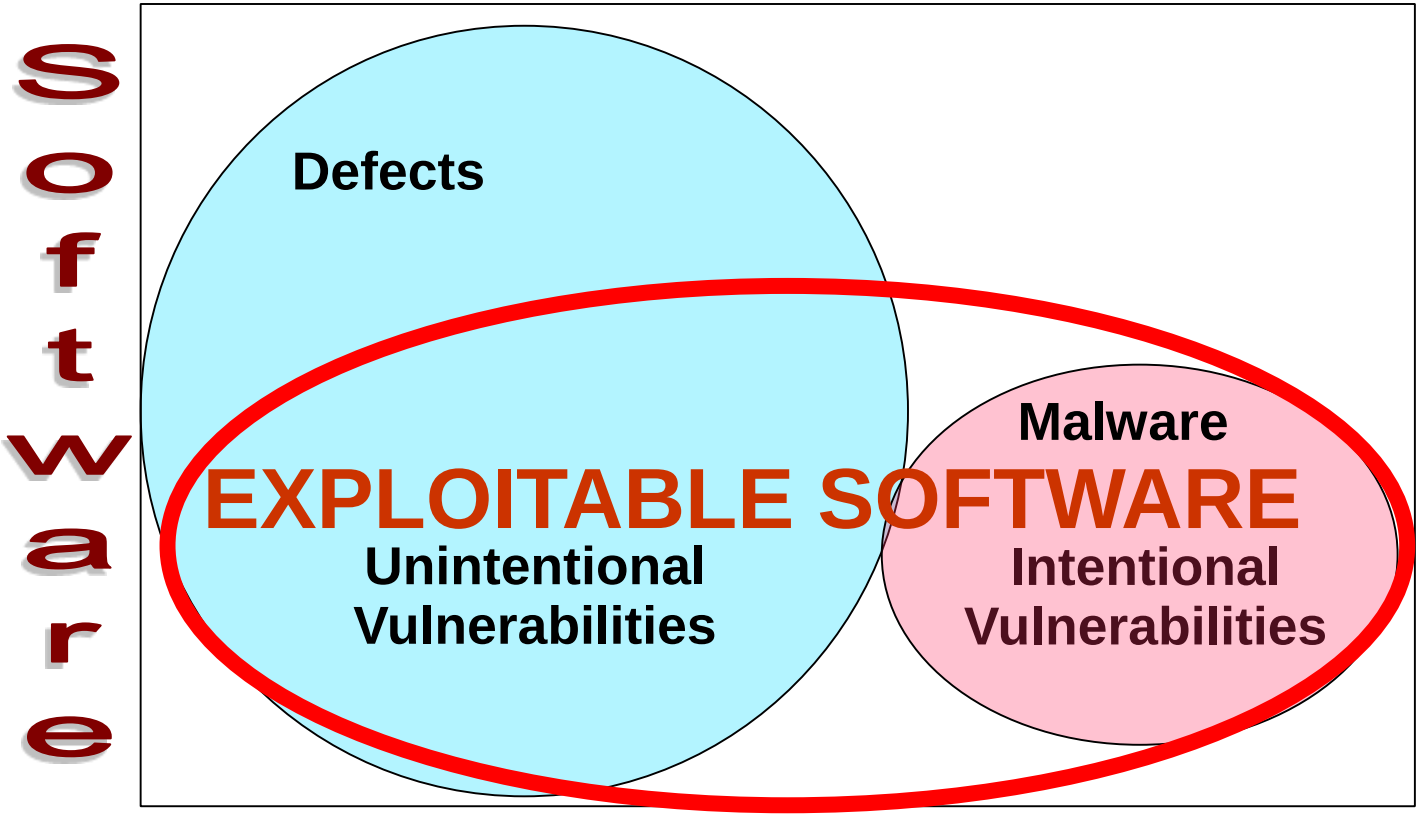
Exploitable Software Weaknesses (CWEs) are exploit targets/vectors for future Zero-Day Attacks

Known weaknesses plague the security threat landscape

- Many of the biggest security risks are issues known about for decades, leaving organizations unnecessarily exposed; organizations must employ fundamental security tactics to address known vulnerabilities to eliminate significant amounts of risk.
  - **44 % of known breaches come from vulnerabilities that are 2-4 years old.**
  - **Server misconfigurations were the number one vulnerability.**
  - **Additional avenues of attack were introduced via connected devices.**
  - **The primary causes of commonly exploited software vulnerabilities are defects, bugs, and logic flaws.**

Software Assurance Addresses Exploitable Software:
Outcomes of non-secure practices and/or malicious intent

Exploitation potential of vulnerability is independent of "intent"



'High quality' can reduce security flaws attributable to defects; yet traditional S/W quality assurance does not address intentional malicious behavior in software

*Intentional vulnerabilities: spyware & malicious logic deliberately imbedded (might not be considered defects)

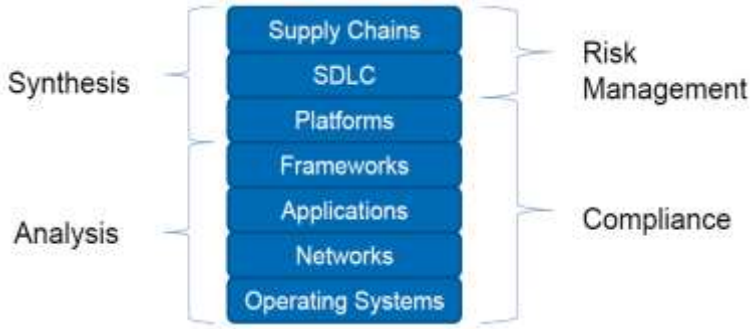
Software Assurance (SwA) is the level of confidence that software functions as intended and is free of vulnerabilities, either intentionally or unintentionally designed or inserted as part of the software throughout the life cycle.*

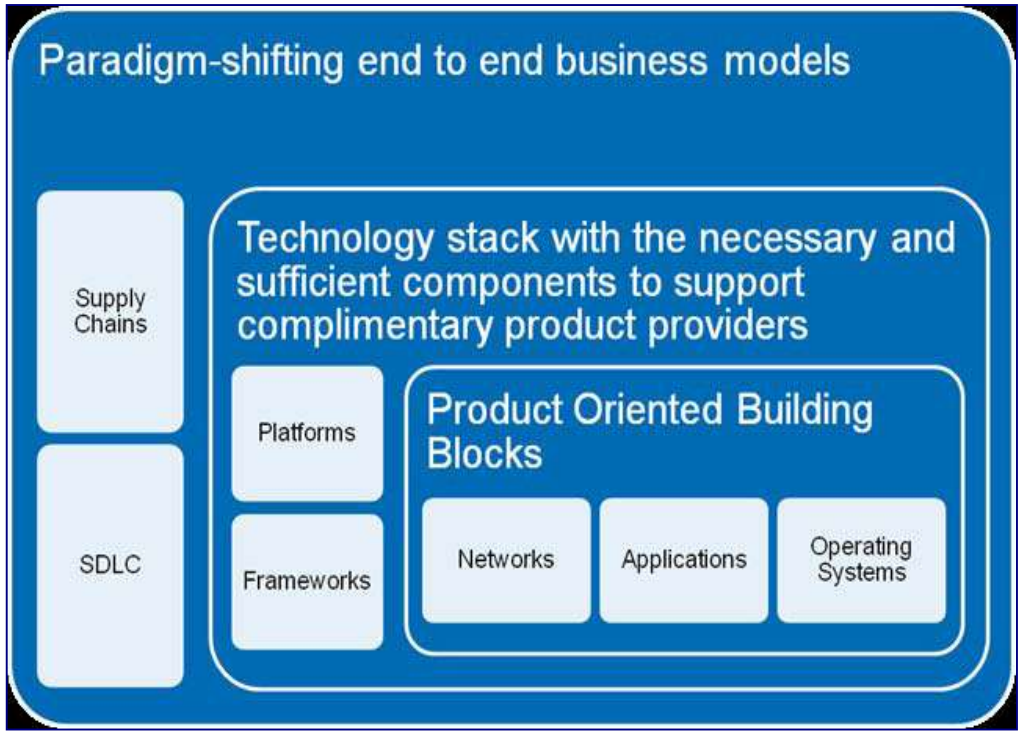
From CNSS Instruction 4009 "National Information Assurance Glossary" (26APR2010)

ICT/software security risk landscape is a convergence between “defense in depth” and “defense in breadth”

Enterprise Risk Management and Governance are security motivators

Acquisition could be considered the beginning of the lifecycle; more than development





Software & Supply Chain Assurance provides a focus for:

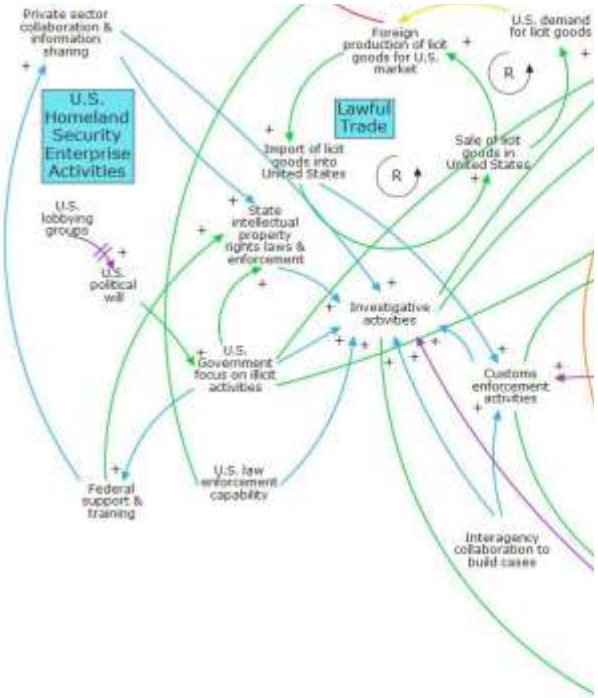
-- Resilient Software and ICT Components,

-- Security in the Component Life Cycle,

-- Cybersecurity in Services, and

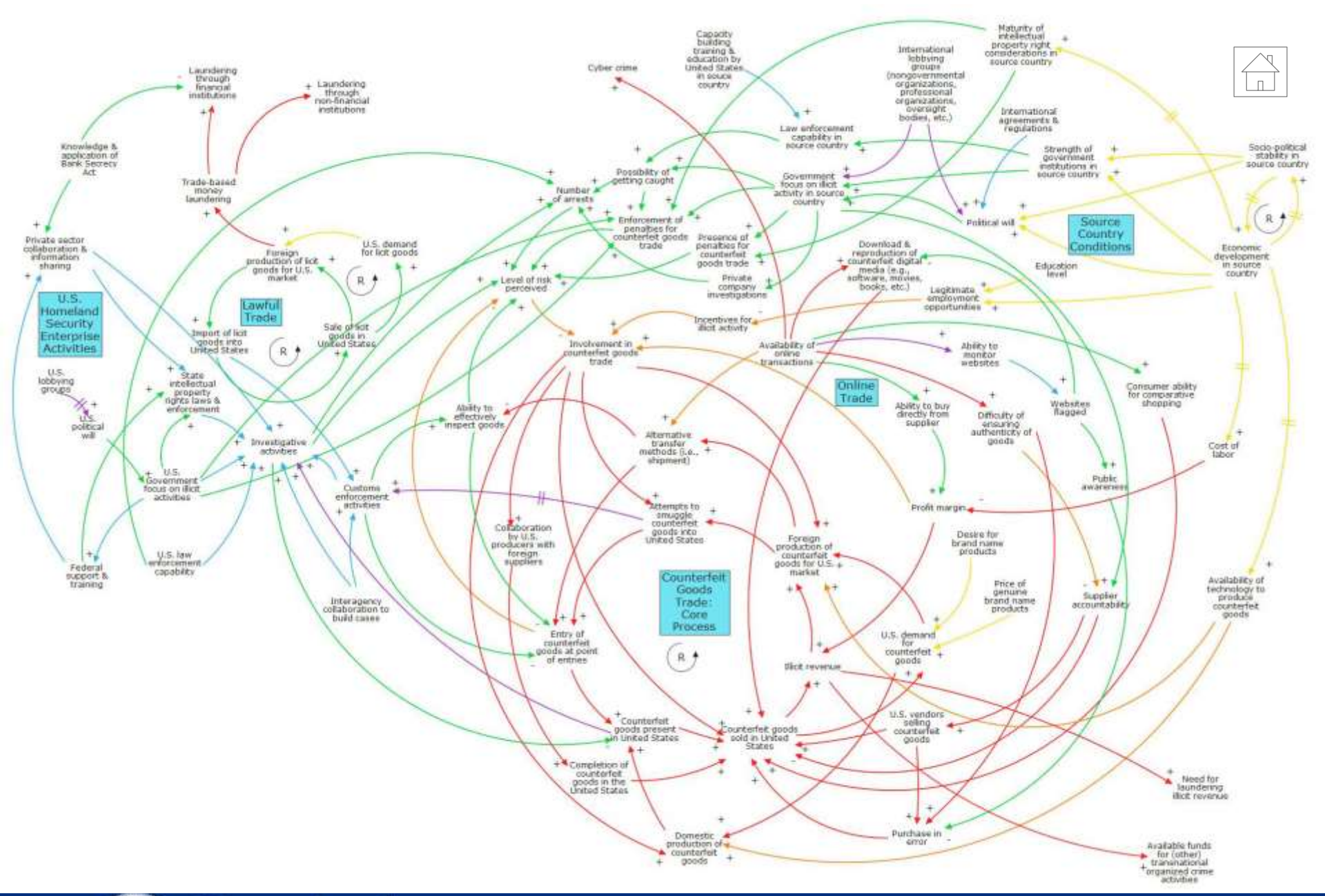
-- Supply Chain Risk Management (mitigating risks of counterfeit & tainted products)

SwA & SCRM Imperative



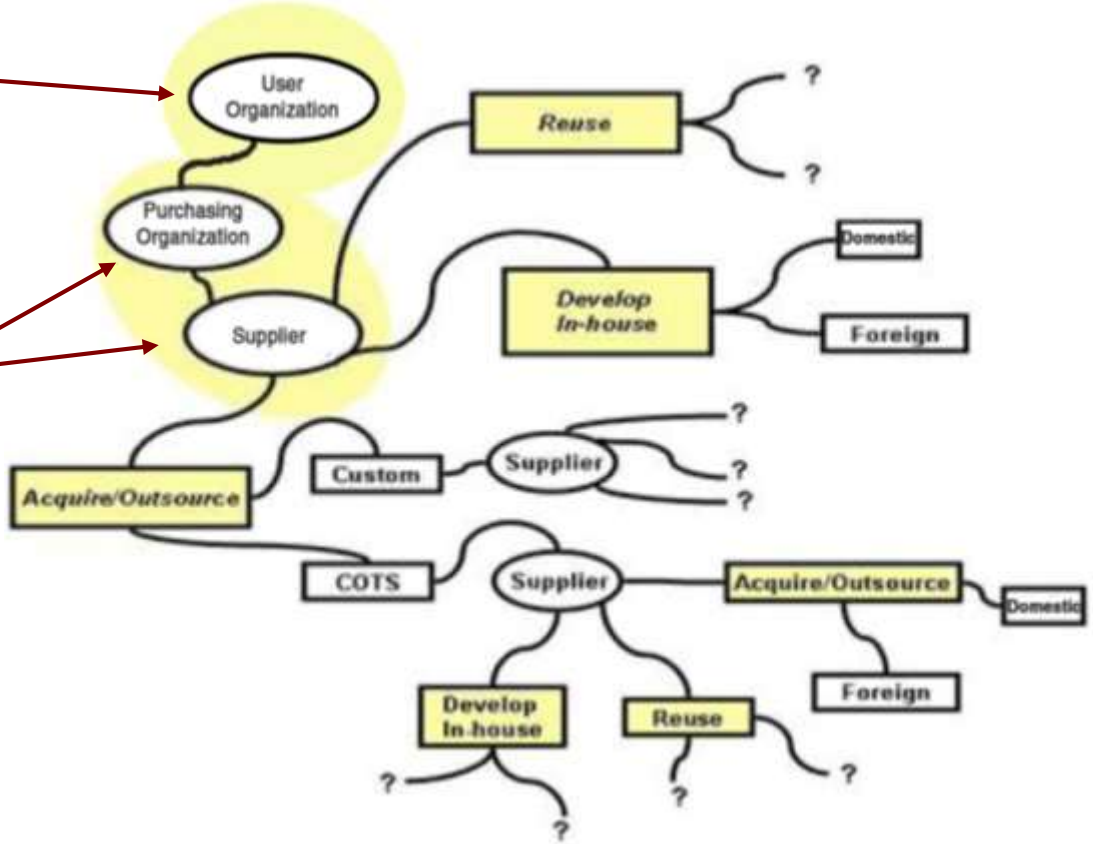
Increased risk from supply chain due to:

- **Increasing dependence on commercial ICT for mission critical systems**
- **Increasing reliance on globally-sourced ICT hardware, software, and services**
  - **Varying levels of development/outsourcing controls**
  - **Lack of transparency in process chain of custody**
  - **Varying levels of acquisition 'due-diligence"**
- **Residual risk passed to end-user enterprise**
  - **Defective and Counterfeit products**
  - **Tainted products with malware, exploitable weaknesses and vulnerabilities**
- **Growing technological sophistication among our adversaries**
  - **Internet enables adversaries to probe, penetrate, and attack us remotely**
  - **Supply chain attacks can exploit products and processes throughout the lifecycle**


Laundering through financial institutions
Laundering through non-financial institutions
Knowledge & application of Bank Secrecy Act
Trade-based money laundering
Private sector collaboration & information sharing
U.S. Homeland Security Enterprise Activities
U.S. lobbying groups
U.S. political will
Federal support & training
U.S. law enforcement capability
Lawful Trade
Foreign production of licit goods for U.S. market
U.S. demand for licit goods
Import of licit goods into United States
Sale of licit goods in United States
State intellectual property rights laws & enforcement
U.S. Government focus on illicit activities
Investigative activities
Customs enforcement activities
Interagency collaboration to build cases
Cyber crime
Capacity building training & education by United States in source country
Law enforcement capability in source country
Possibility of getting caught
Number of arrests
Enforcement of penalties for counterfeit goods trade
Presence of penalties for counterfeit goods trade
Private company investigations
Level of risk perceived
Government focus on illicit activity in source country
International lobbying groups (nongovernmental organizations, professional organizations, oversight bodies, etc.)
Maturity of intellectual property right considerations in source country
International agreements & regulations
Strength of government institutions in source country
Socio-political stability in source country
Political will
Source Country Conditions
Economic development in source country
Education level
Download & reproduction of counterfeit digital media (e.g., software, movies, books, etc.)
Legitimate employment opportunities
Incentives for illicit activity
Involvement in counterfeit goods trade
Availability of online transactions
Ability to monitor websites
Online Trade
Ability to buy directly from supplier
Difficulty of ensuring authenticity of goods
Websites flagged
Consumer ability for comparative shopping
Cost of labor
Public awareness
Ability to effectively inspect goods
Alternative transfer methods (i.e., shipment)
Attempts to smuggle counterfeit goods into United States
Collaboration by U.S. producers with foreign suppliers
Profit margin
Foreign production of counterfeit goods for U.S. market
Desire for brand name products
Price of genuine brand name products
Supplier accountability
Availability of technology to produce counterfeit goods
Counterfeit Goods Trade: Core Process
Entry of counterfeit goods at point of entries
U.S. demand for counterfeit goods
Illicit revenue
Counterfeit goods present in United States
Counterfeit goods sold in United States
U.S. vendors selling counterfeit goods
Completion of counterfeit goods in the United States
Need for laundering illicit revenue
Domestic production of counterfeit goods
Purchase in error
Available funds for (other) transnational organized crime activities

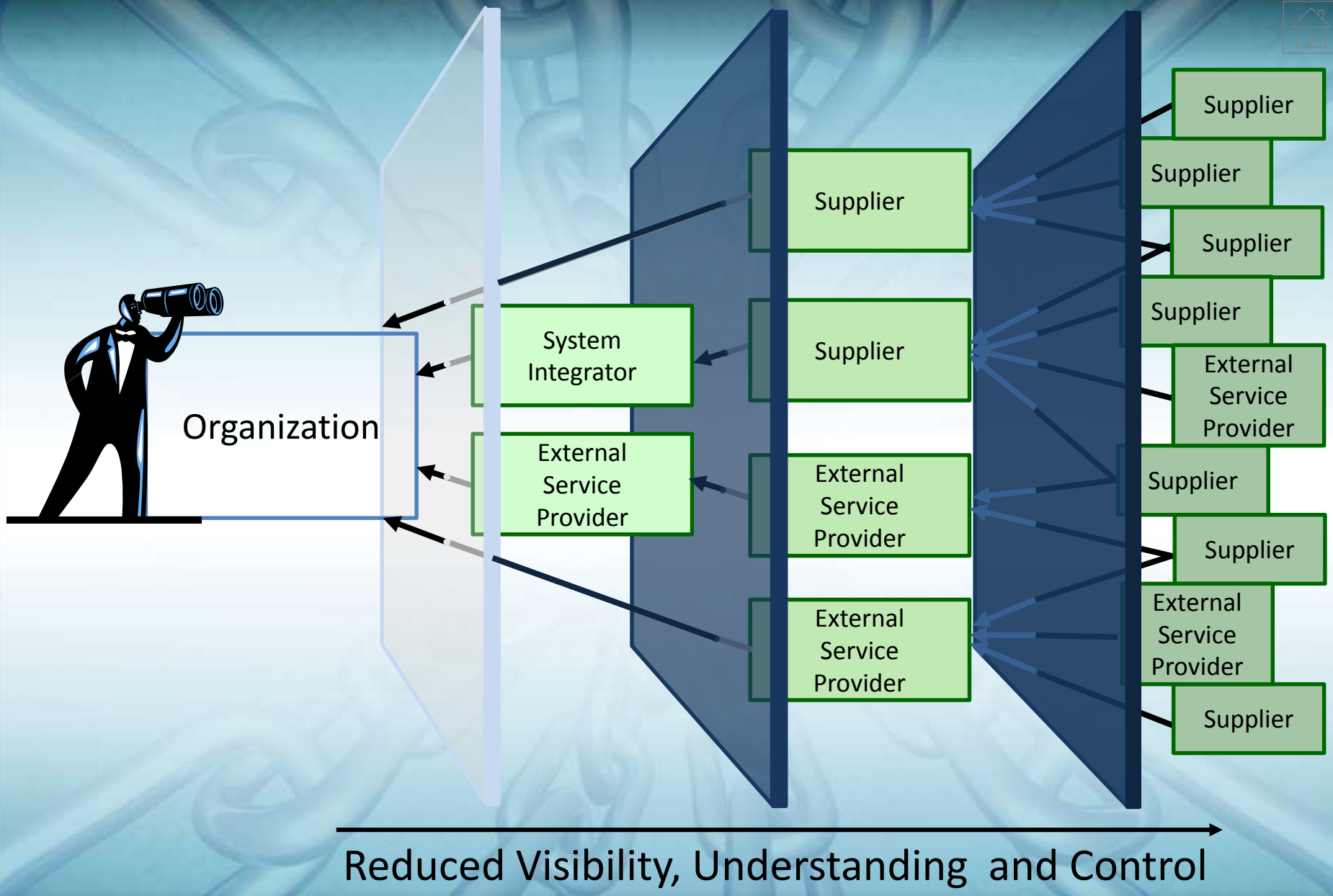
Risk Management (Enterprise ⟷ Project):
Shared Processes & Practices ⟷ Different Focuses

- Enterprise-Level:
  - Regulatory compliance
  - Changing threat environment
  - Business Case
- Program/Project-Level:
  - Cost
  - Schedule
  - Performance



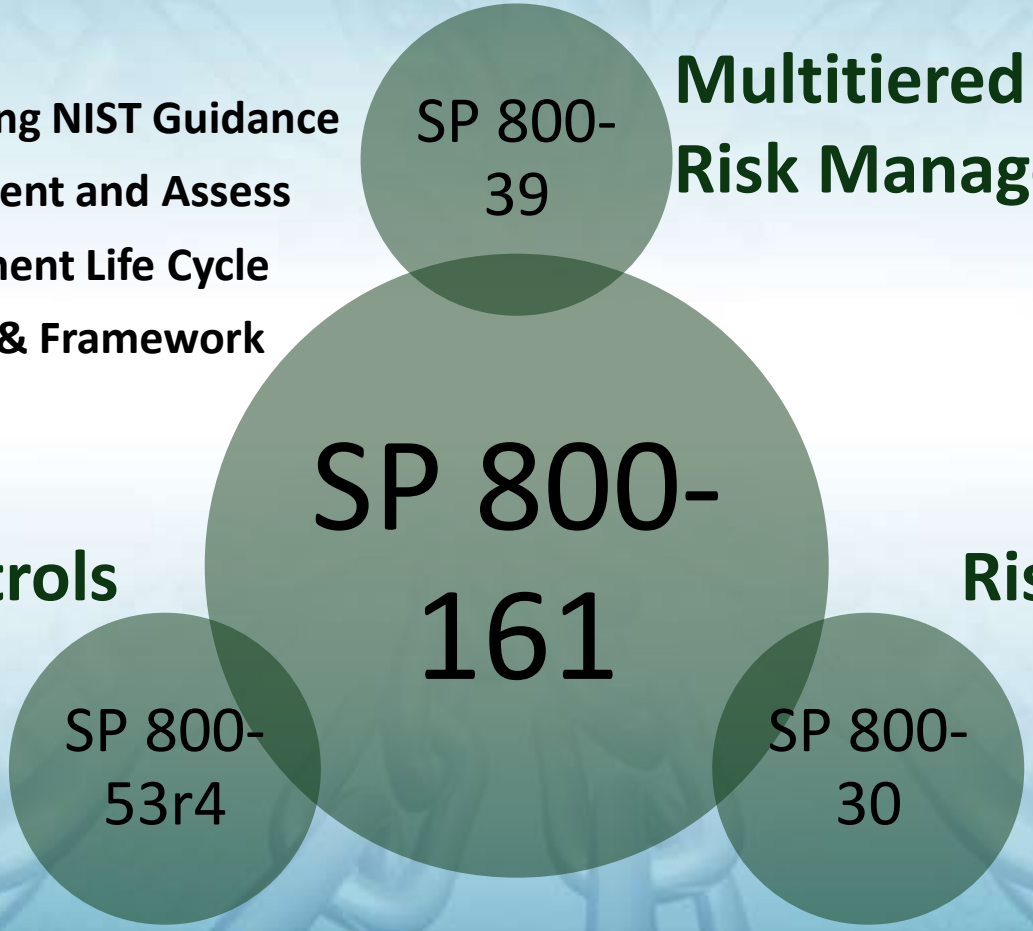
Who makes risk decisions?
Who determines 'fitness for use' for 'technically acceptable' criteria?
Who "owns" residual risk from tainted/counterfeit products?

* "Tainted" products are those that are corrupted with malware, or exploitable weaknesses & vulnerabilities that put users at risk


Organization
System Integrator
External Service Provider
Supplier
Supplier
External Service Provider
External Service Provider
Supplier
Supplier
Supplier
External Service Provider
Supplier
Supplier
External Service Provider
Supplier
Reduced Visibility, Understanding and Control

SP 800-161, Supply Chain Risk Management for Federal Information Systems and Organizations

- **Building on existing NIST Guidance**
- **Ability to Implement and Assess**
- **System Development Life Cycle**
- **Threat Scenarios & Framework**
- **ICT SCRM Plan**



Multitiered Organizational Risk Management

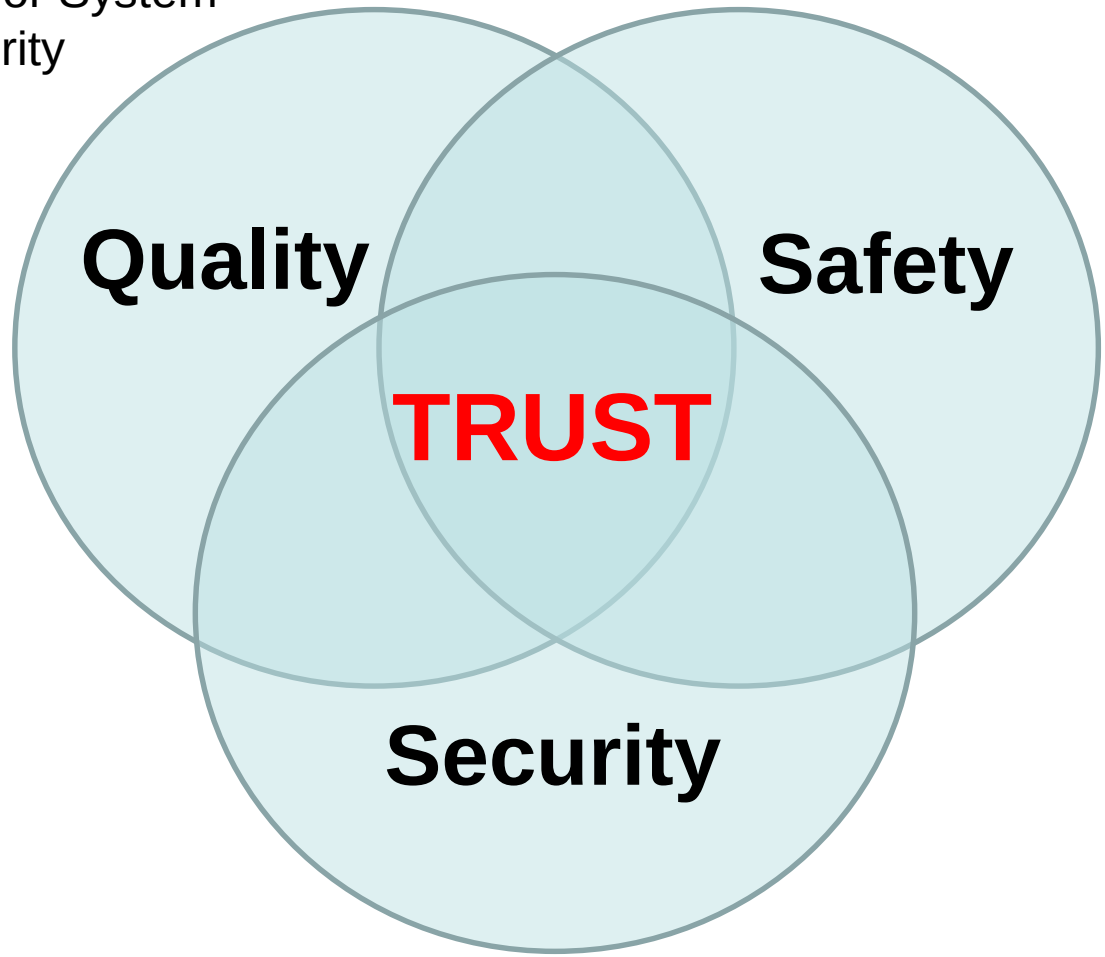
Security Controls

Risk Assessment

Assurance relative to Trust

Managing Effects of Unintentional Defects in Component or System Integrity

Managing Consequences of Unintentional Defects



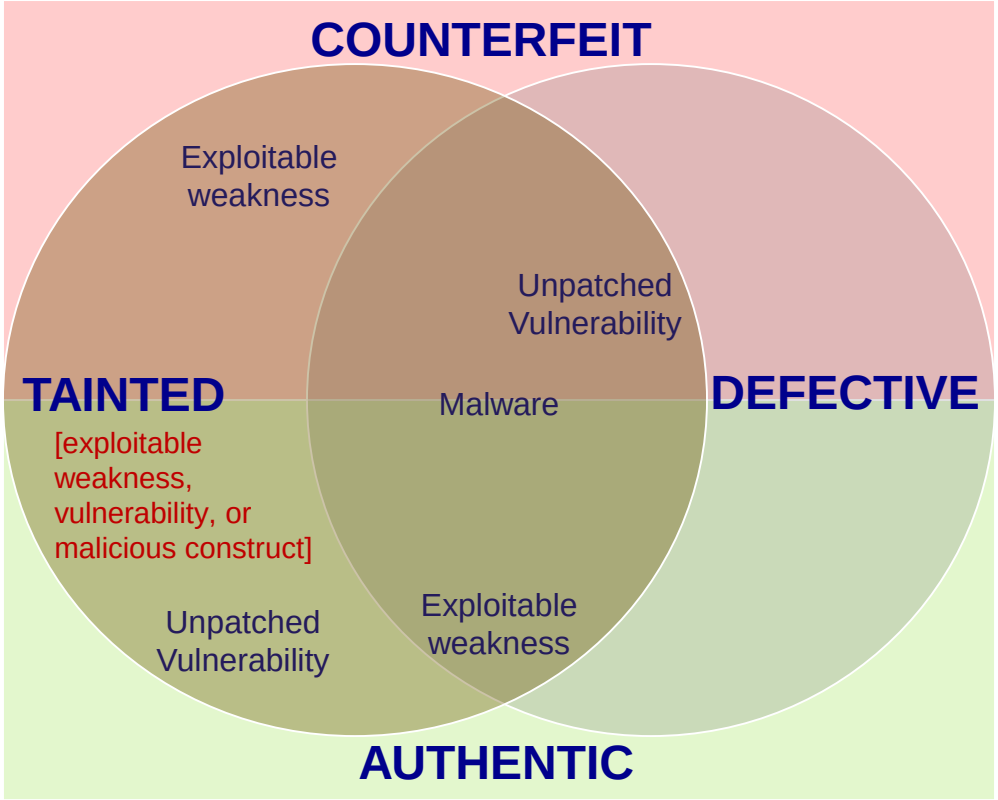
Managing Consequences of Intentional Exploitable Constructs

SSCA Focus on Tainted Components

Mitigating risks attributable to exploitable non-conforming constructs in ICT

"Tainted" products are those that are corrupted with malware, or exploitable weaknesses & vulnerabilities that put users at risk

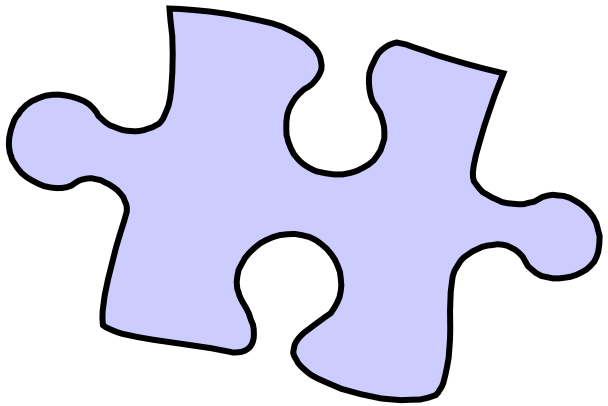
- Enable 'scalable' detection and reporting of tainted ICT components
  - Leverage/mature related existing standardization efforts
  - Provide Taxonomies, schema & structured representations with defined observables & indicators for conveying information:
    - Tainted constructs:
      - Malicious logic/malware (MAEC),
      - Exploitable Weaknesses (CWE);
      - Vulnerabilities (CVE)
    - Attack Patterns (CAPEC)
- Catalogue Diagnostic Methods, Controls, Countermeasures, & Mitigation Practices

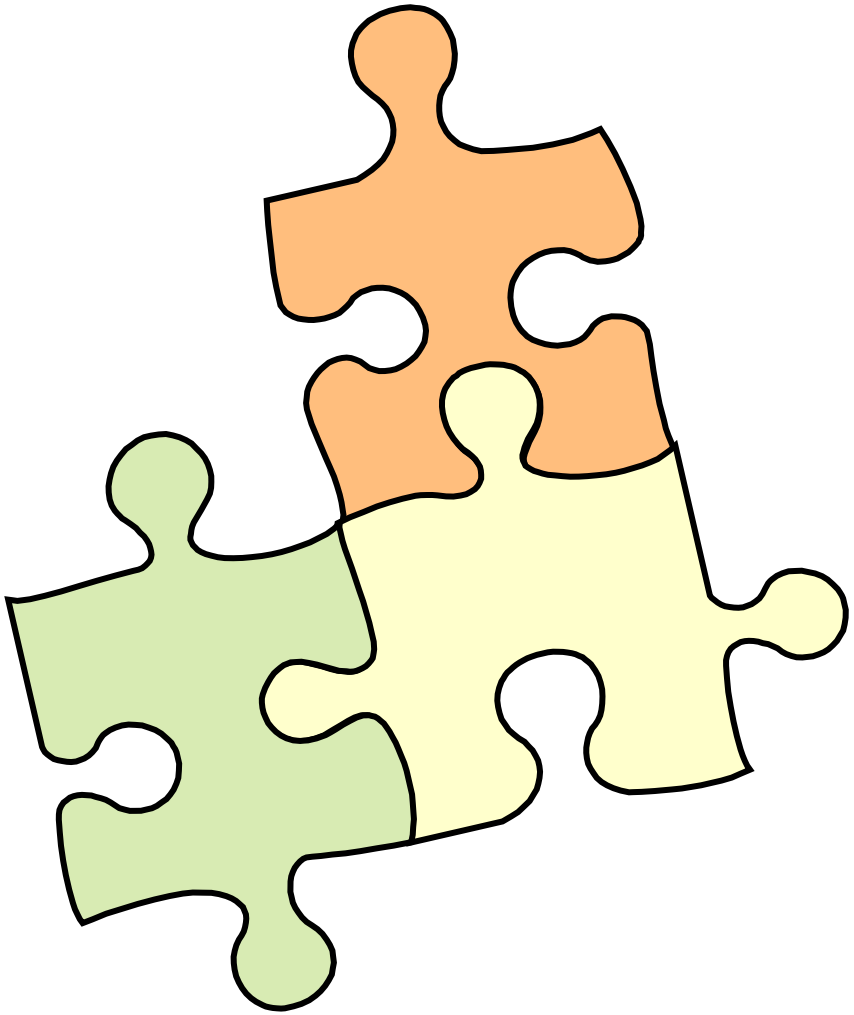


Components can become tainted intentionally or unintentionally throughout the supply chain, SDLC, and in Ops & sustainment

*Text demonstrates *examples* of overlap

Automation is *one piece*





of the SwA & SCRM puzzle.

“Making Security Measureable”:
measurablesecurity.mitre.org

Sponsored by DHS CS&C
with MITRE as technical lead

Resources provided for
voluntary adoption

Open, community efforts that
are *free* to use

XML-based

Software Assurance

Software Assurance (SwA) is the level of confidence that software functions as intended and is free of vulnerabilities, either intentionally or unintentionally designed or inserted as part of the software throughout the life cycle.*

Derived From: CNSSI-4009

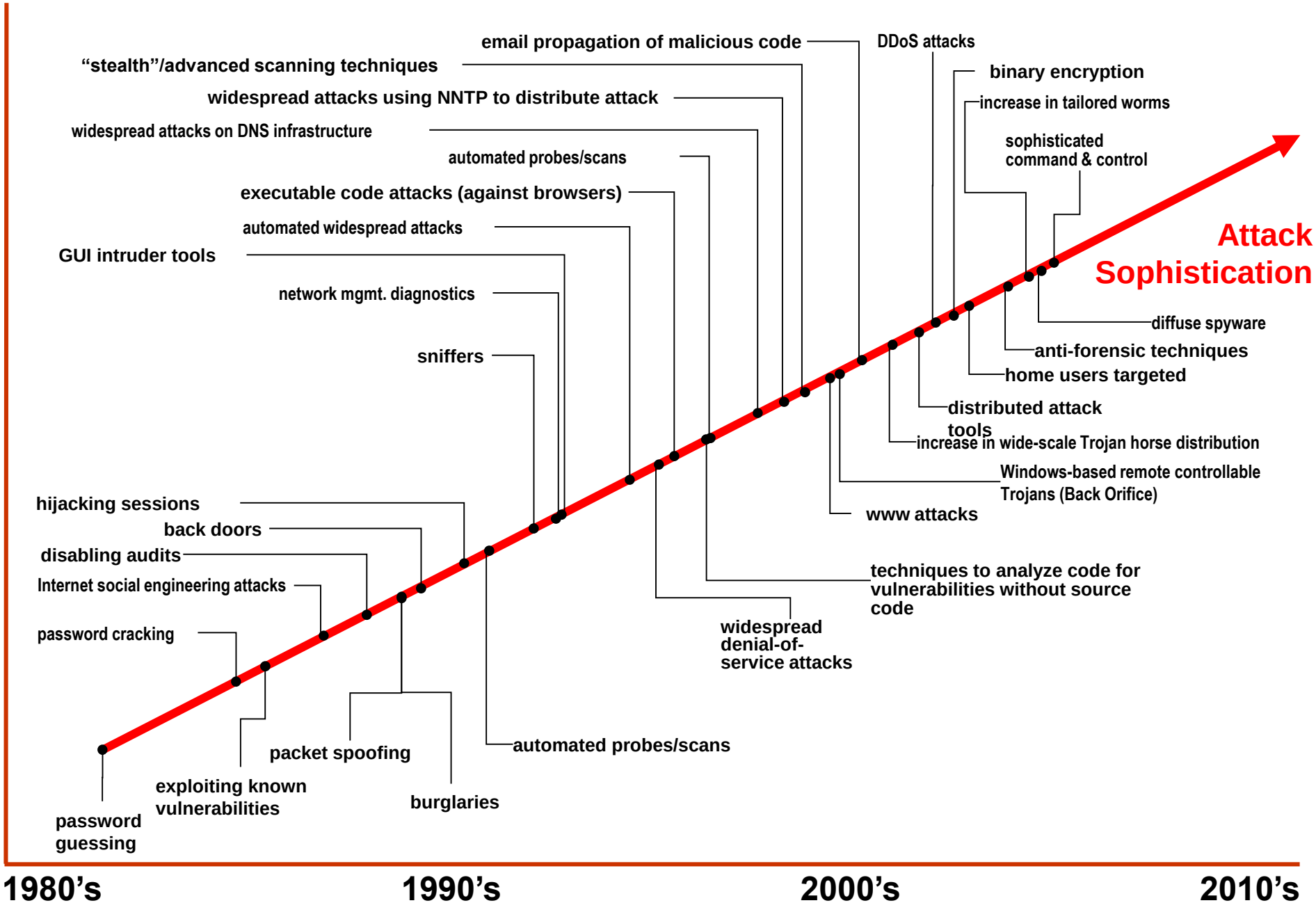
Automation

Languages, enumerations,
registries, tools, and repositories

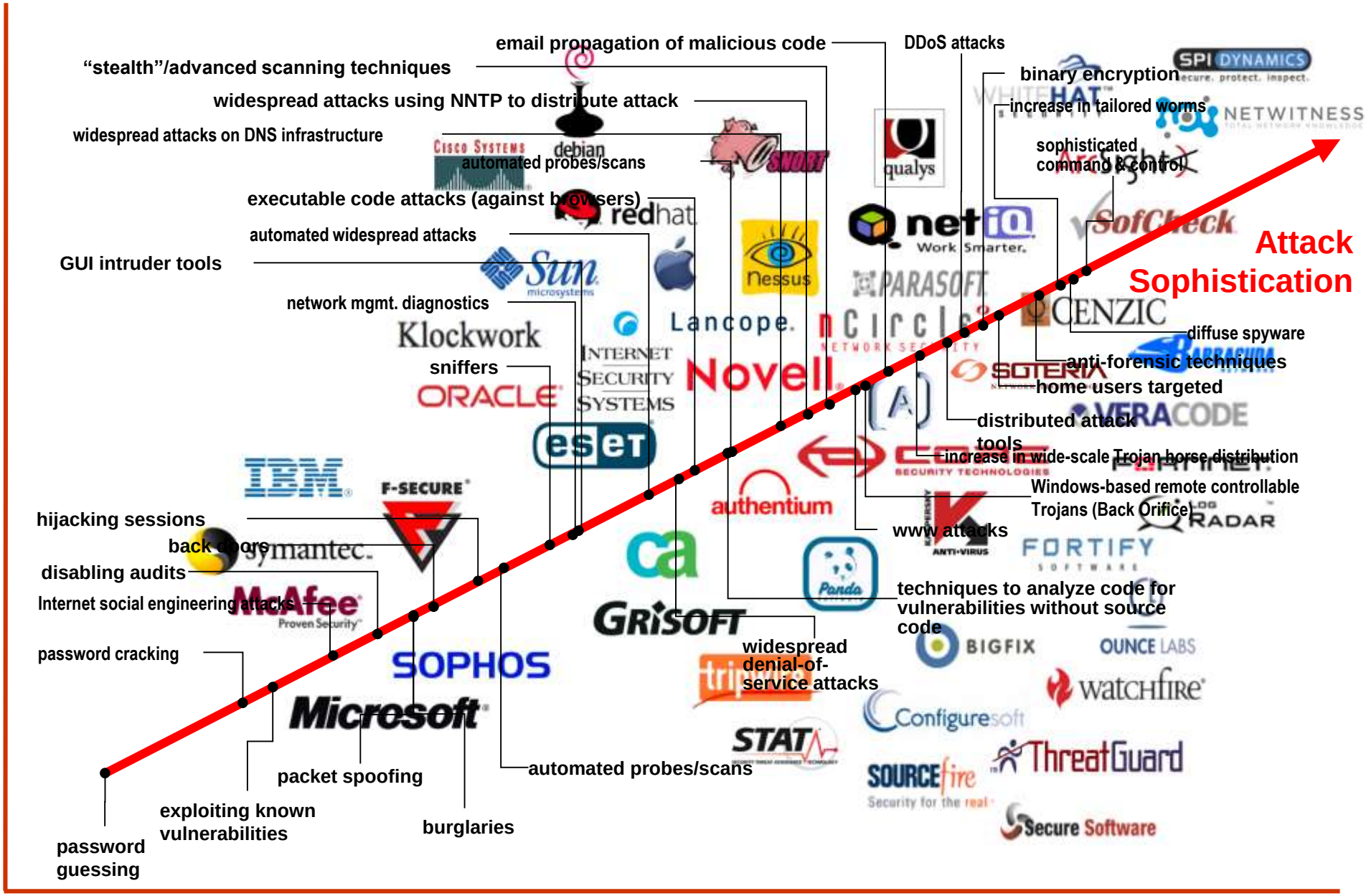
throughout the Lifecycle

Including design, coding, testing, deployment, configuration and operation

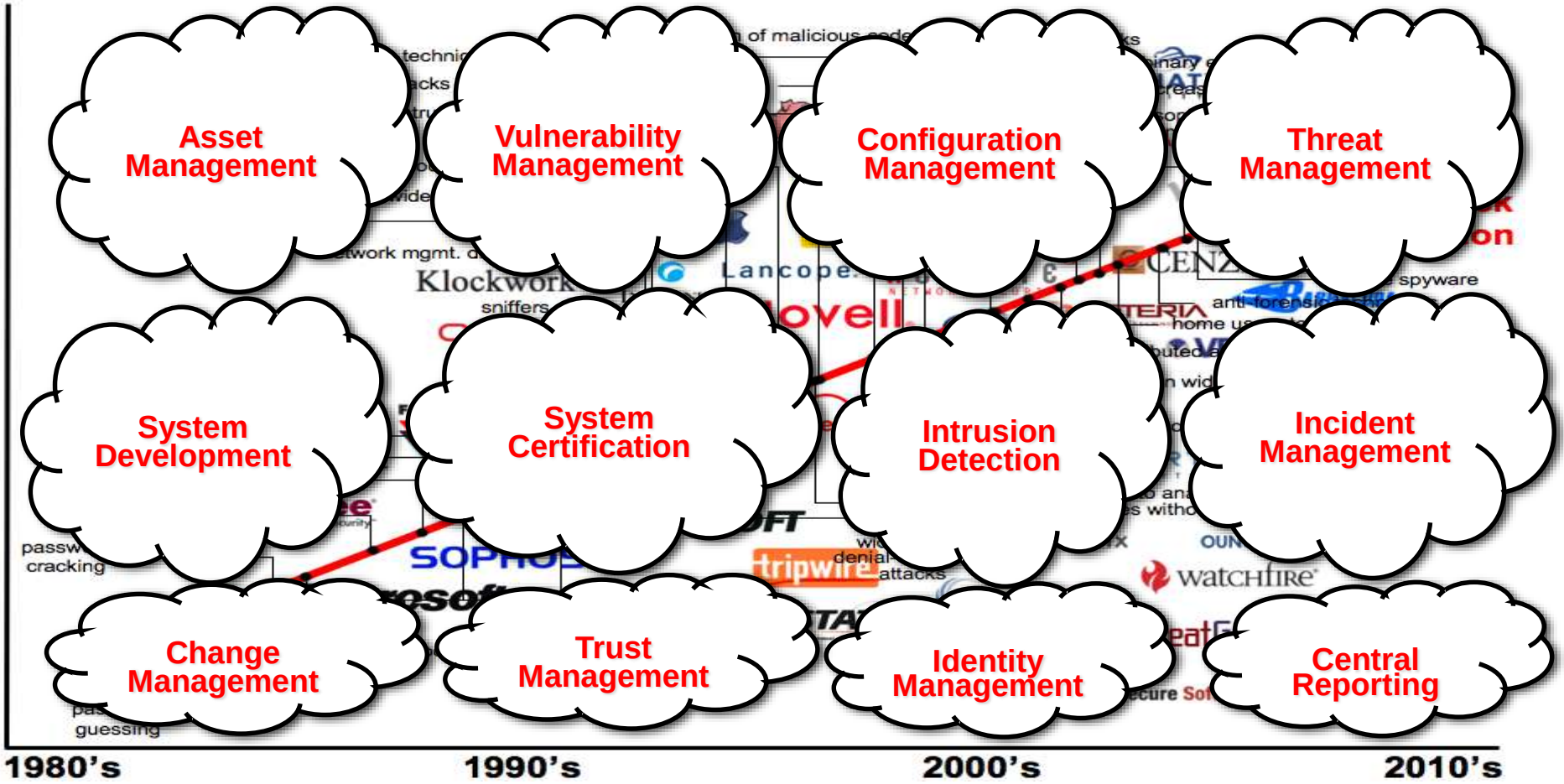
Cyber Threats Emerged Over Time



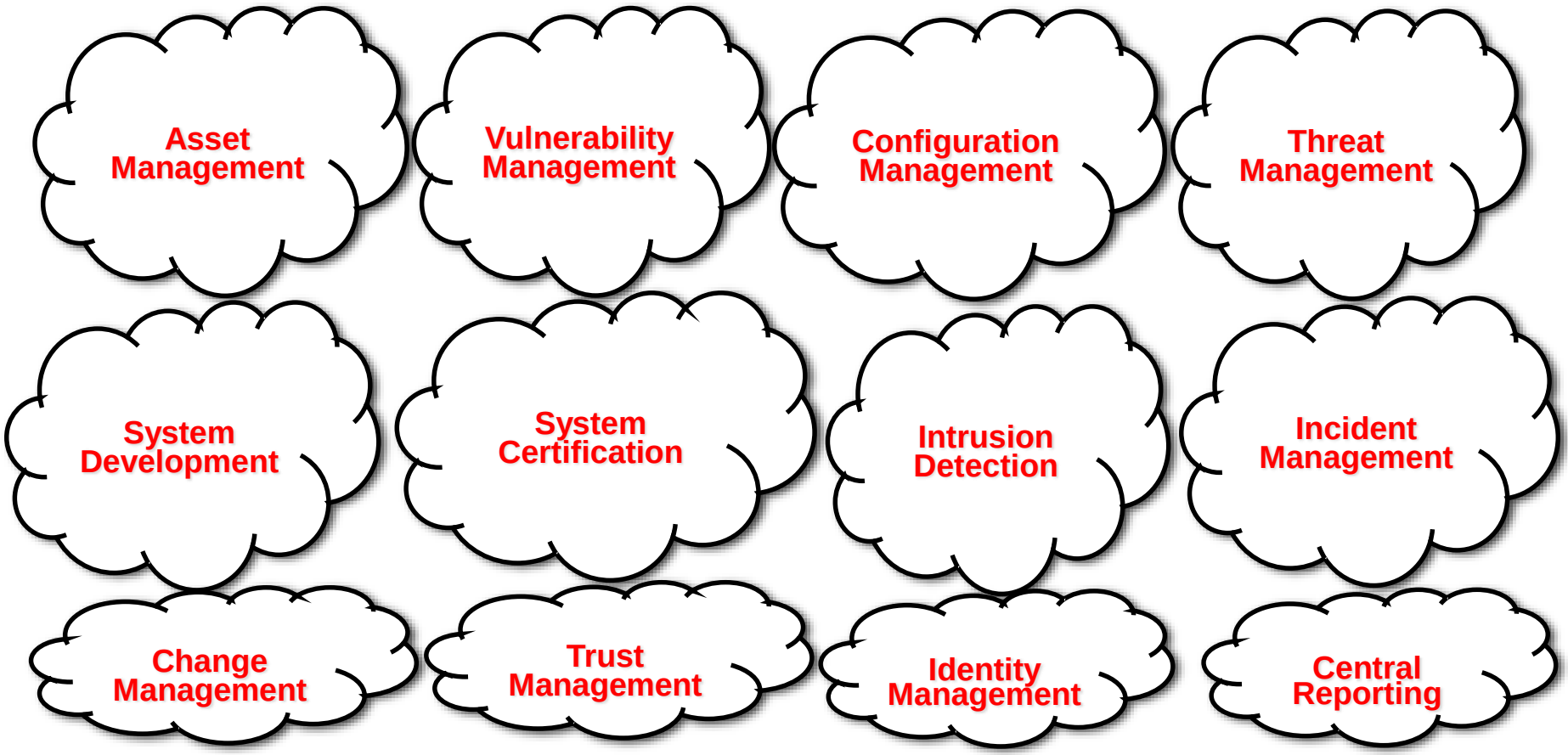
Solutions Also Emerged Over Time



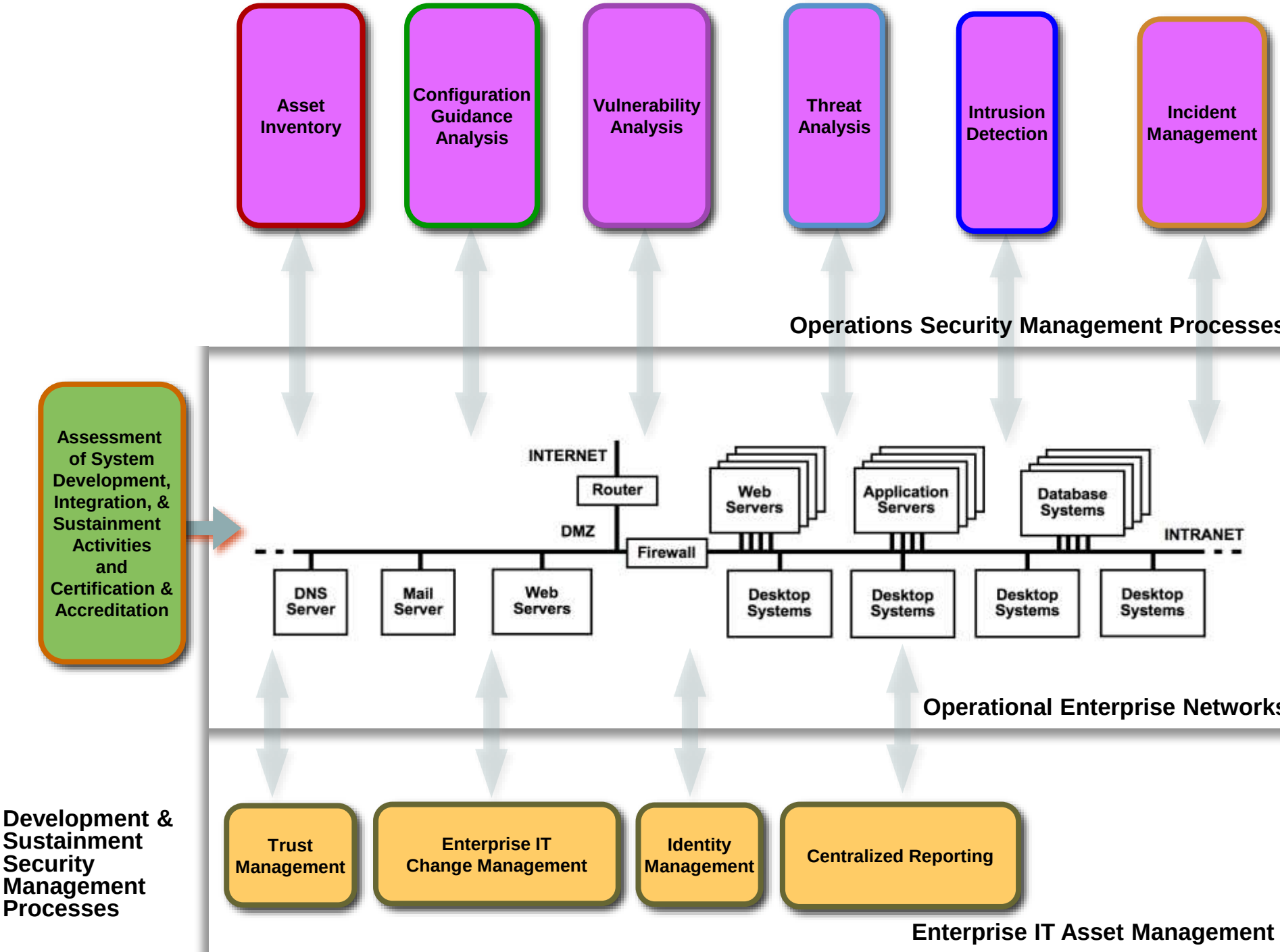
Architecting Security with Information Standards for COIs

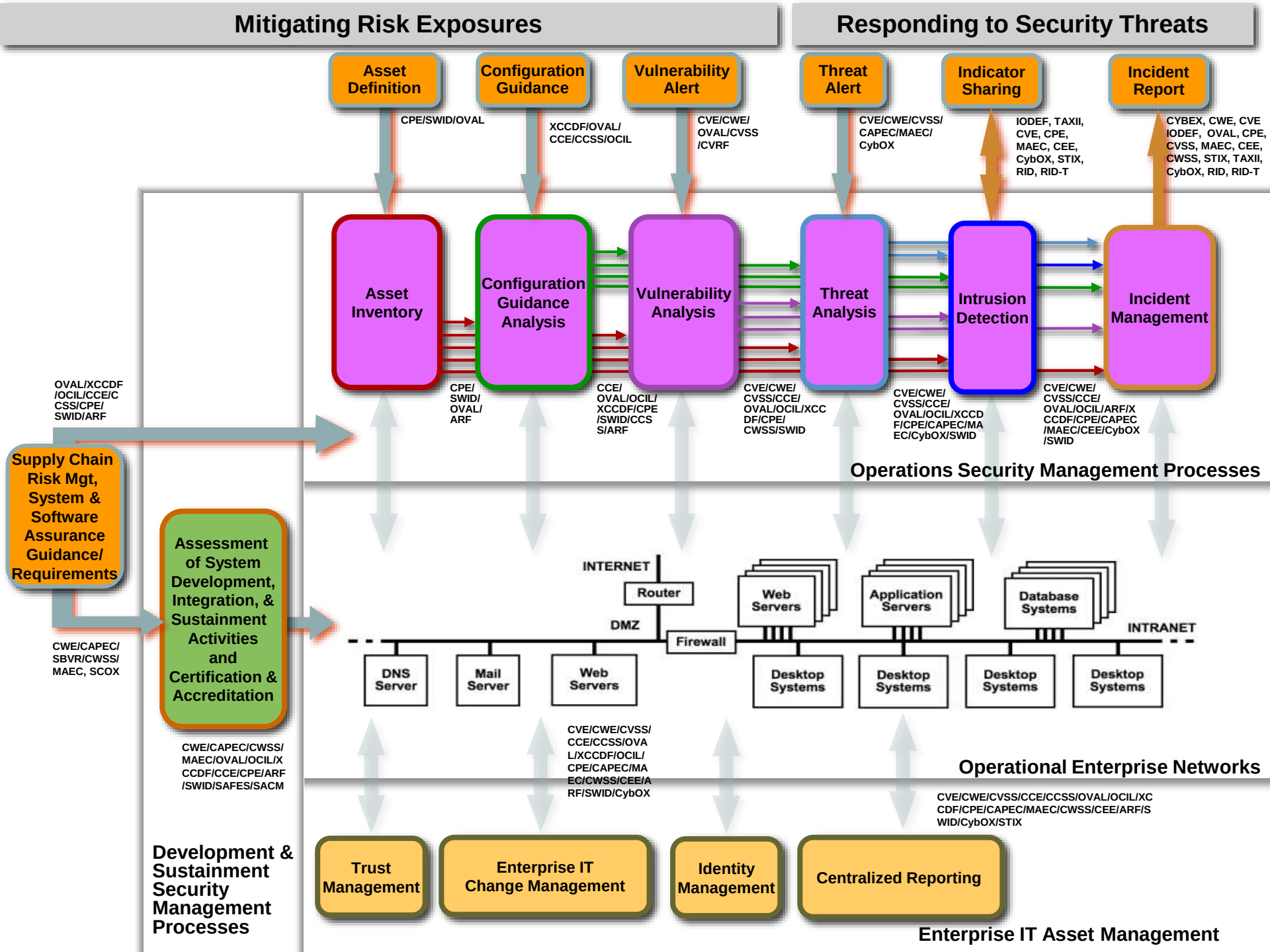


Making Security Measurable™



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


Mitigating Risk Exposures
Responding to Security Threats
Asset Definition
Configuration Guidance
Vulnerability Alert
Threat Alert
Indicator Sharing
Incident Report
CPE/SWID/OVAL
XCCDF/OVAL/ CCE/CCSS/OCIL
CVE/CWE/ OVAL/CVSS /CVRF
CVE/CWE/CVSS/ CAPEC/MAEC/ CybOX
IODEF, TAXII, CVE, CPE, MAEC, CEE, CybOX, STIX, RID, RID-T
CYBEX, CWE, CVE IODEF, OVAL, CPE, CVSS, MAEC, CEE, CWSS, STIX, TAXII, CybOX, RID, RID-T
Asset Inventory
Configuration Guidance Analysis
Vulnerability Analysis
Threat Analysis
Intrusion Detection
Incident Management
OVAL/XCCDF /OCIL/CCE/C CSS/CPE/ SWID/ARF
CPE/ SWID/ OVAL/ ARF
CCE/ OVAL/OCIL/ XCCDF/CPE /SWID/CCS S/ARF
CVE/CWE/ CVSS/CCE/ OVAL/OCIL/XCC DF/CPE/ CWSS/SWID
CVE/CWE/ CVSS/CCE/ OVAL/OCIL/XCCD F/CPE/CAPEC/MA EC/CybOX/SWID
CVE/CWE/ CVSS/CCE/ OVAL/OCIL/ARF/X CCDF/CPE/CAPEC /MAEC/CEE/CybOX /SWID
Operations Security Management Processes
Supply Chain Risk Mgt, System & Software Assurance Guidance/ Requirements
Assessment of System Development, Integration, & Sustainment Activities and Certification & Accreditation
CWE/CAPEC/ SBVR/CWSS/ MAEC, SCOX
CWE/CAPEC/CWSS/ MAEC/OVAL/OCIL/X CCDF/CCE/CPE/ARF /SWID/SAFES/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
CVE/CWE/CVSS/ CCE/CCSS/OVA L/XCCDF/OCIL/ CPE/CAPEC/MA EC/CWSS/CEE/A RF/SWID/CybOX
Operational Enterprise Networks
CVE/CWE/CVSS/CCE/CCSS/OVAL/OCIL/XC CDF/CPE/CAPEC/MAEC/CWSS/CEE/ARF/S WID/CybOX/STIX
Development & Sustainment Security Management Processes
Trust Management
Enterprise IT Change Management
Identity Management
Centralized Reporting
Enterprise IT Asset Management

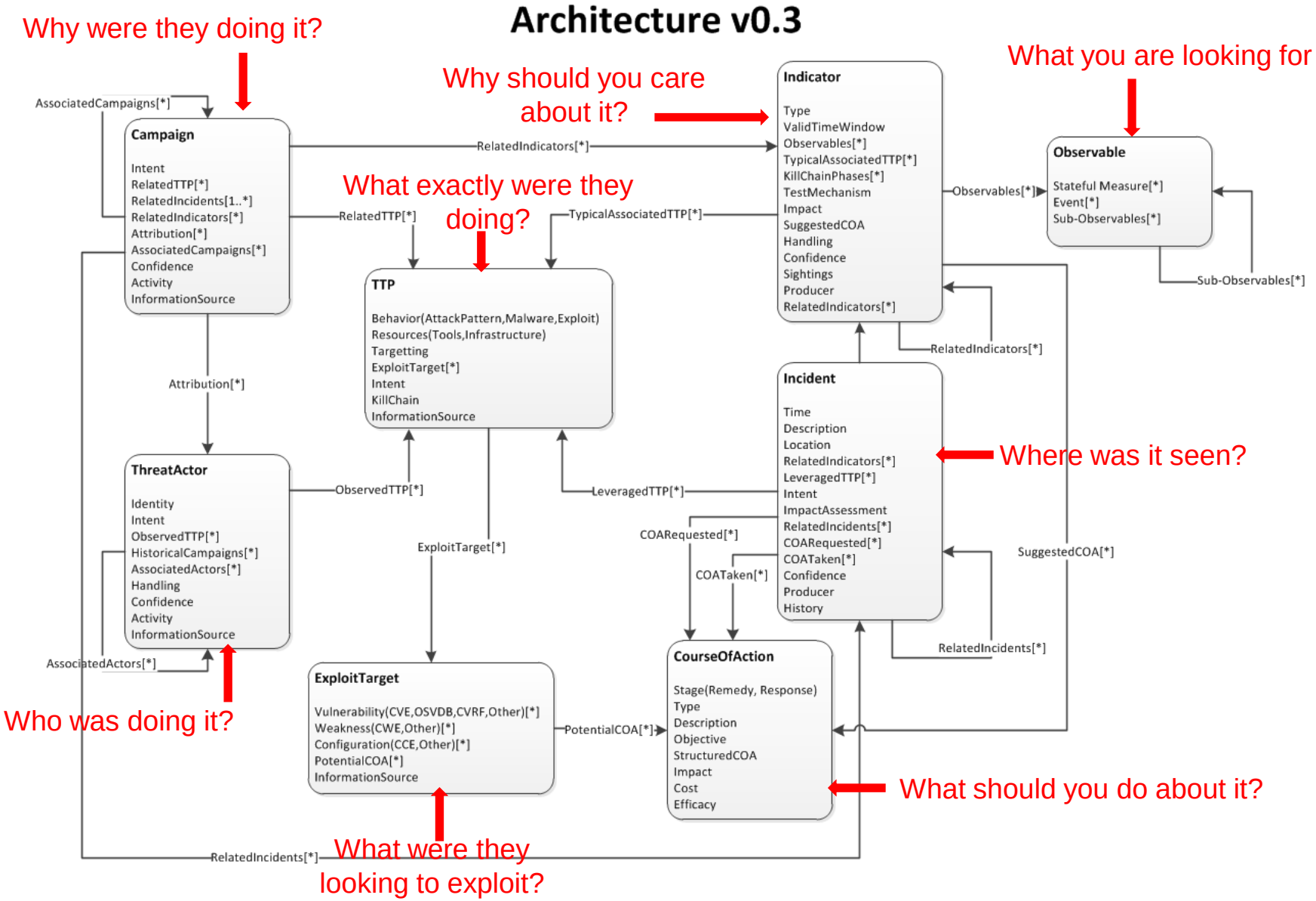
Cyber Ecosystem Standardization Efforts

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 ensure 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)
How can I share threat information?	• STIX (Structure Threat Information)
What events should be logged, and how?	• CEE (Events)
How can I aggregate assessment results?	- ARF (Assessment Results)

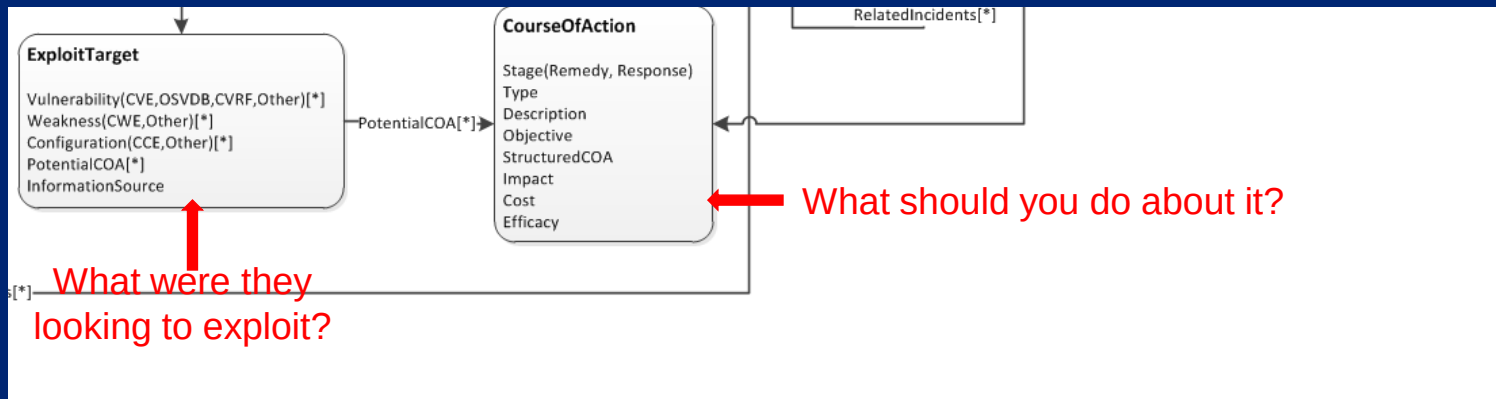
- Many standards are XML-based; enabling automation of information exchange
- Several standards support multiple enterprise cybersecurity functions

Kill Chain – Exploit Targets – Courses of Action

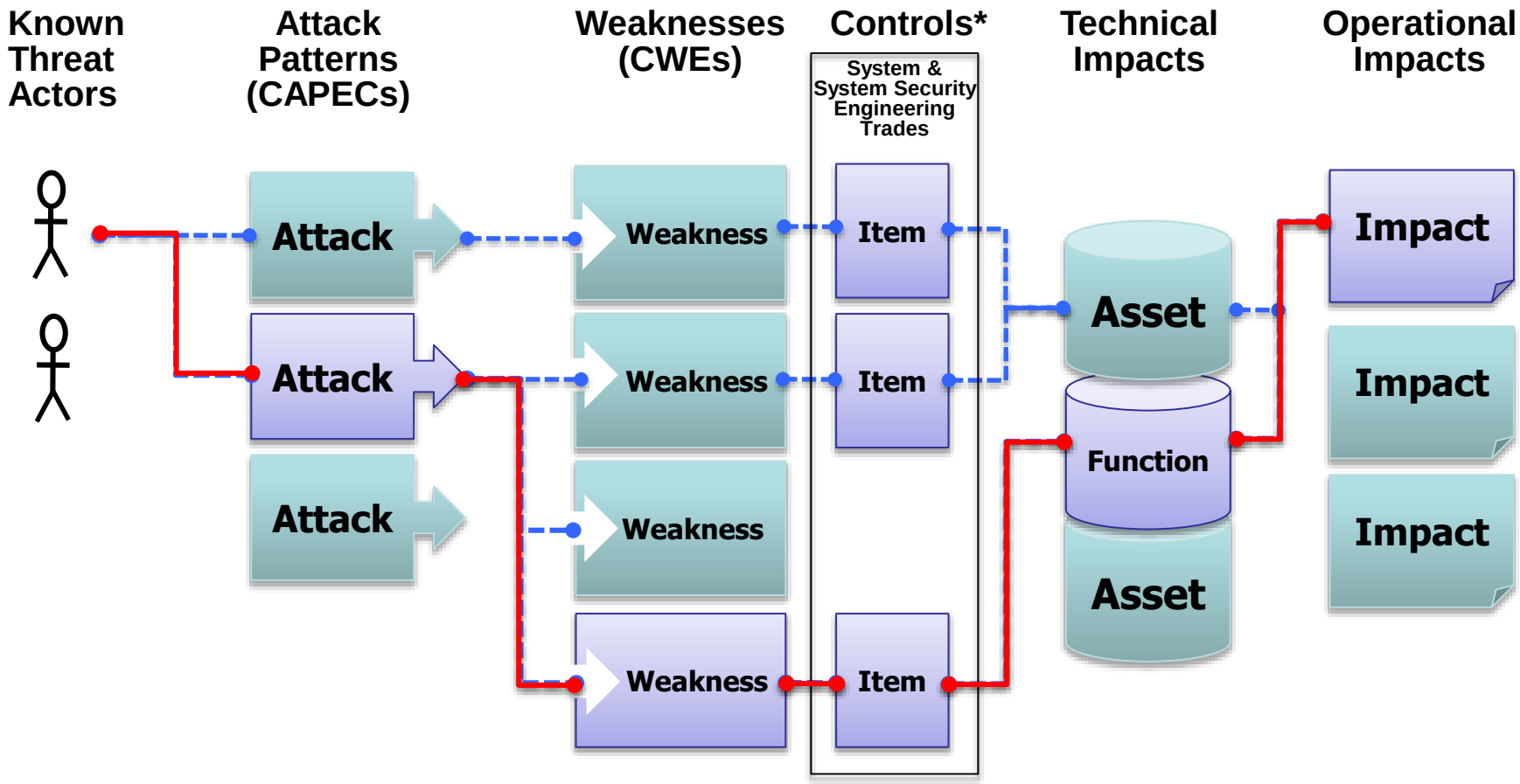
Structured Threat Information eXpression (STIX) Architecture v0.3



What could/should have been done to harden the attack surface/vector to prevent the target from being exploitable?

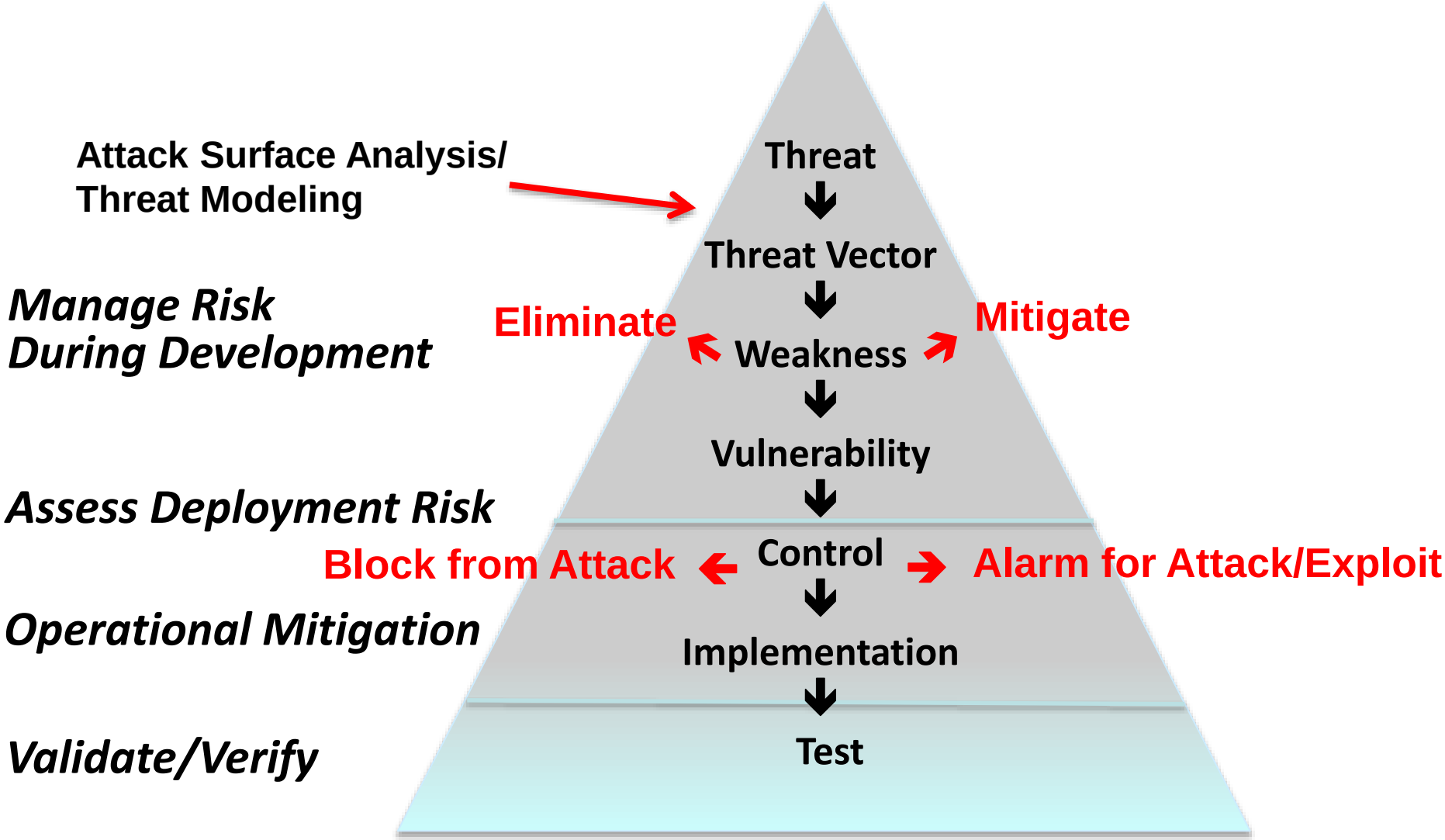


Assurance: Mitigating Attacks That Impact Operations

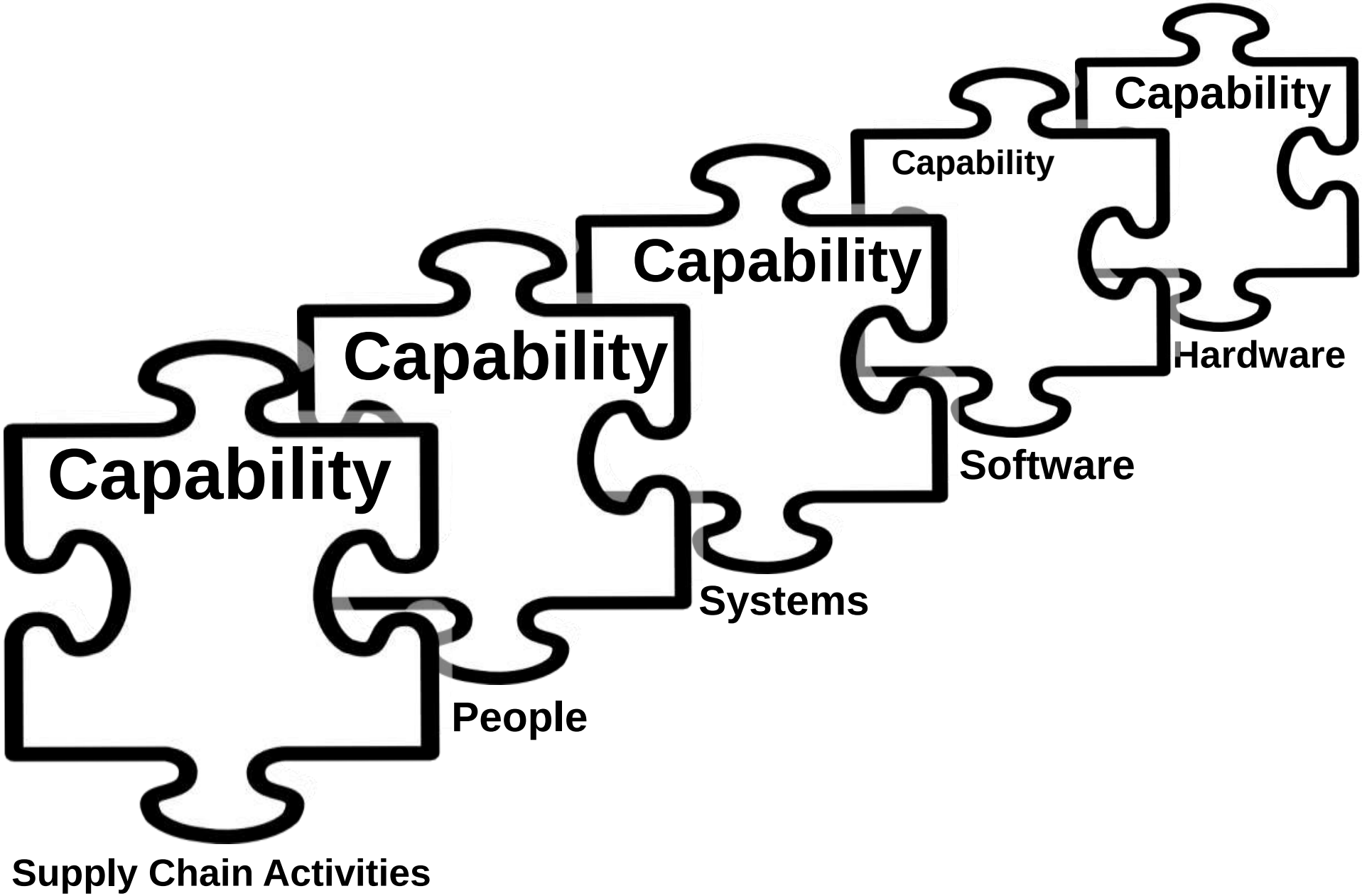


* Controls include architecture choices, design choices, added security functions, activities & processes, physical decomposition choices, code assessments, design reviews, dynamic testing, and pen testing

Assurance on the Management of Weaknesses



Many Capabilities Support the Mission



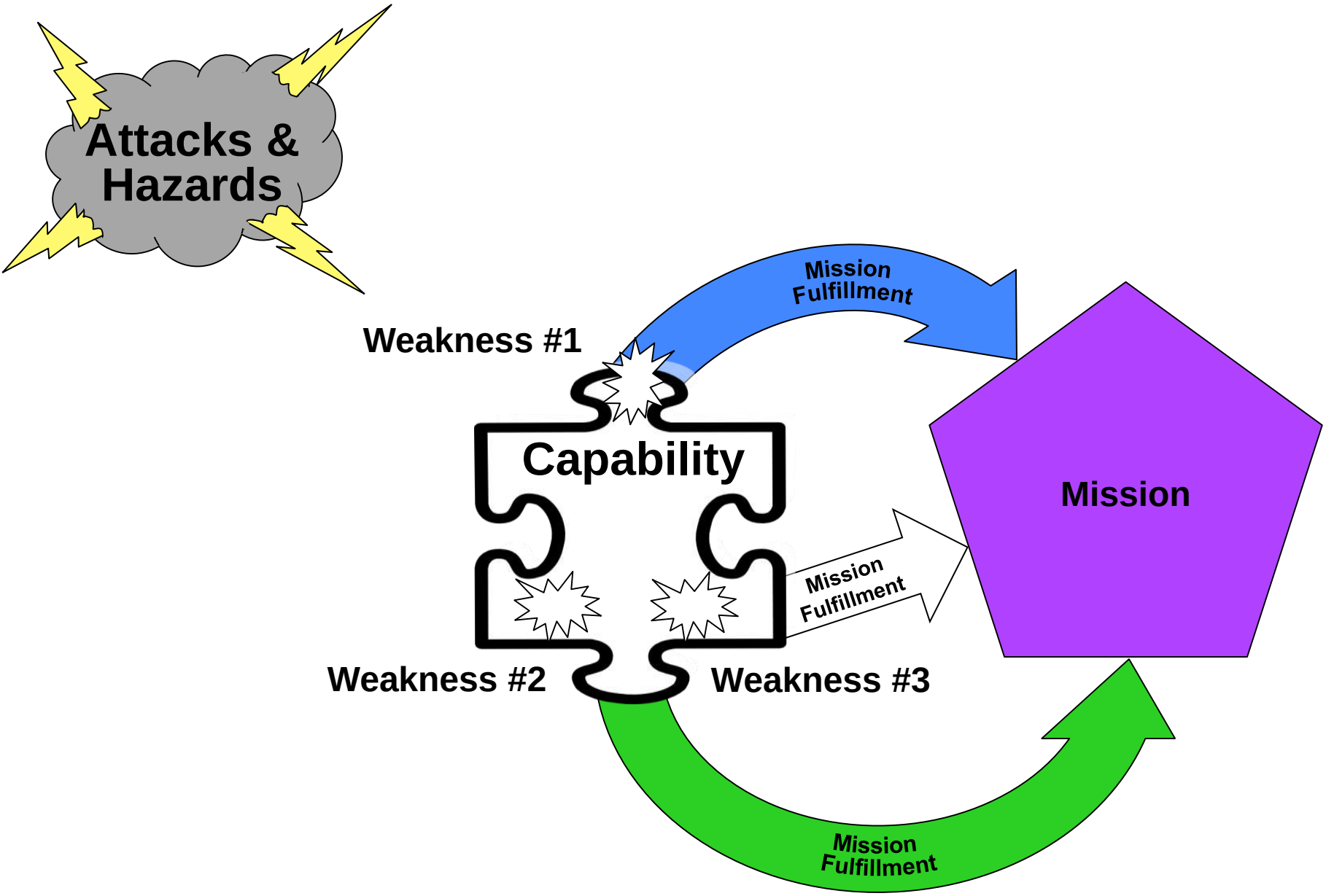
“Weaknesses” address “observables” that might be exploited:

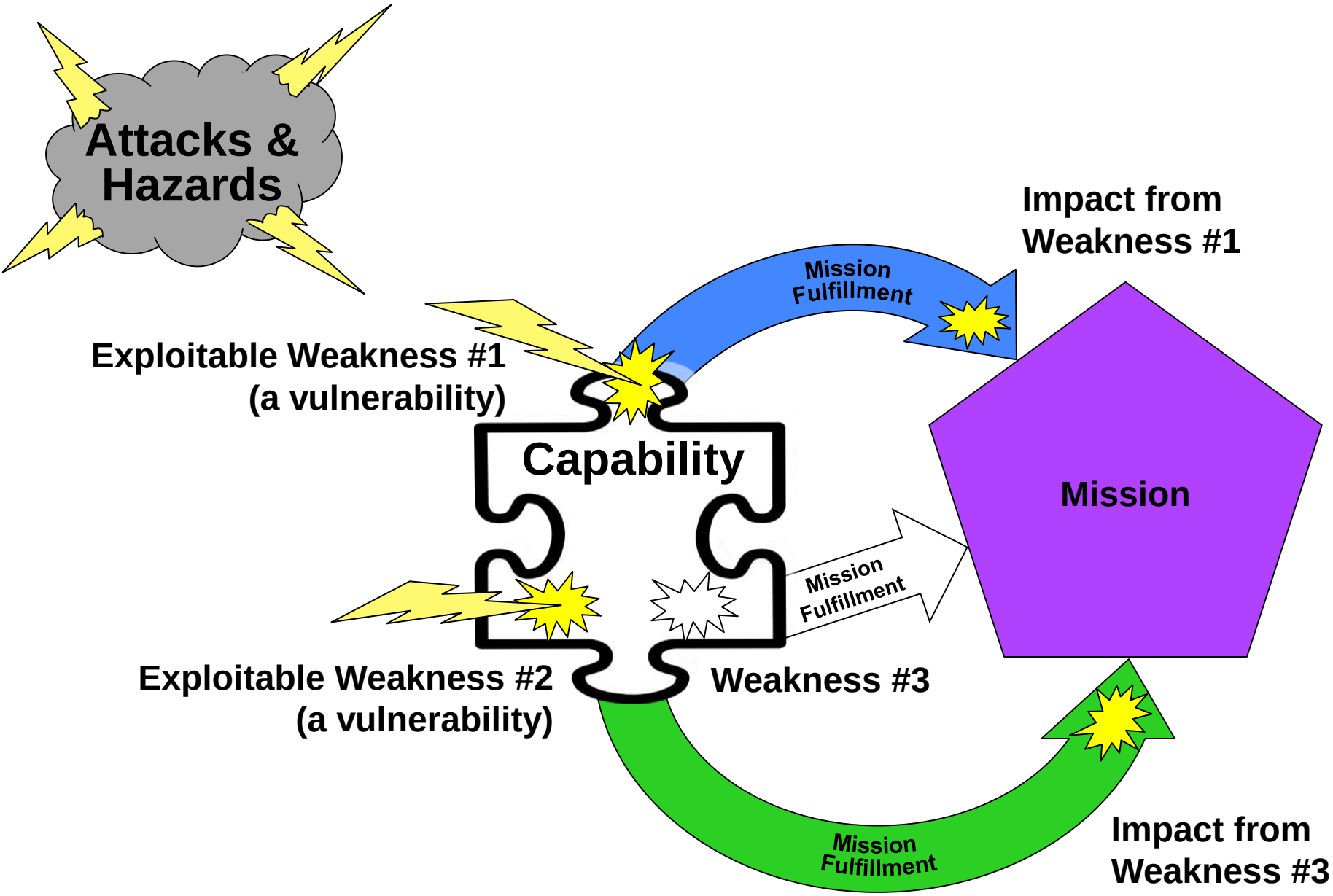
A *weakness* is a property of software, hardware, system (including design and architecture), or process/practice or behavior that, under right conditions, may permit unintended/unauthorized action.

*Common Weakness Enumeration (CWE) v3 http://cwe.mitre.org/

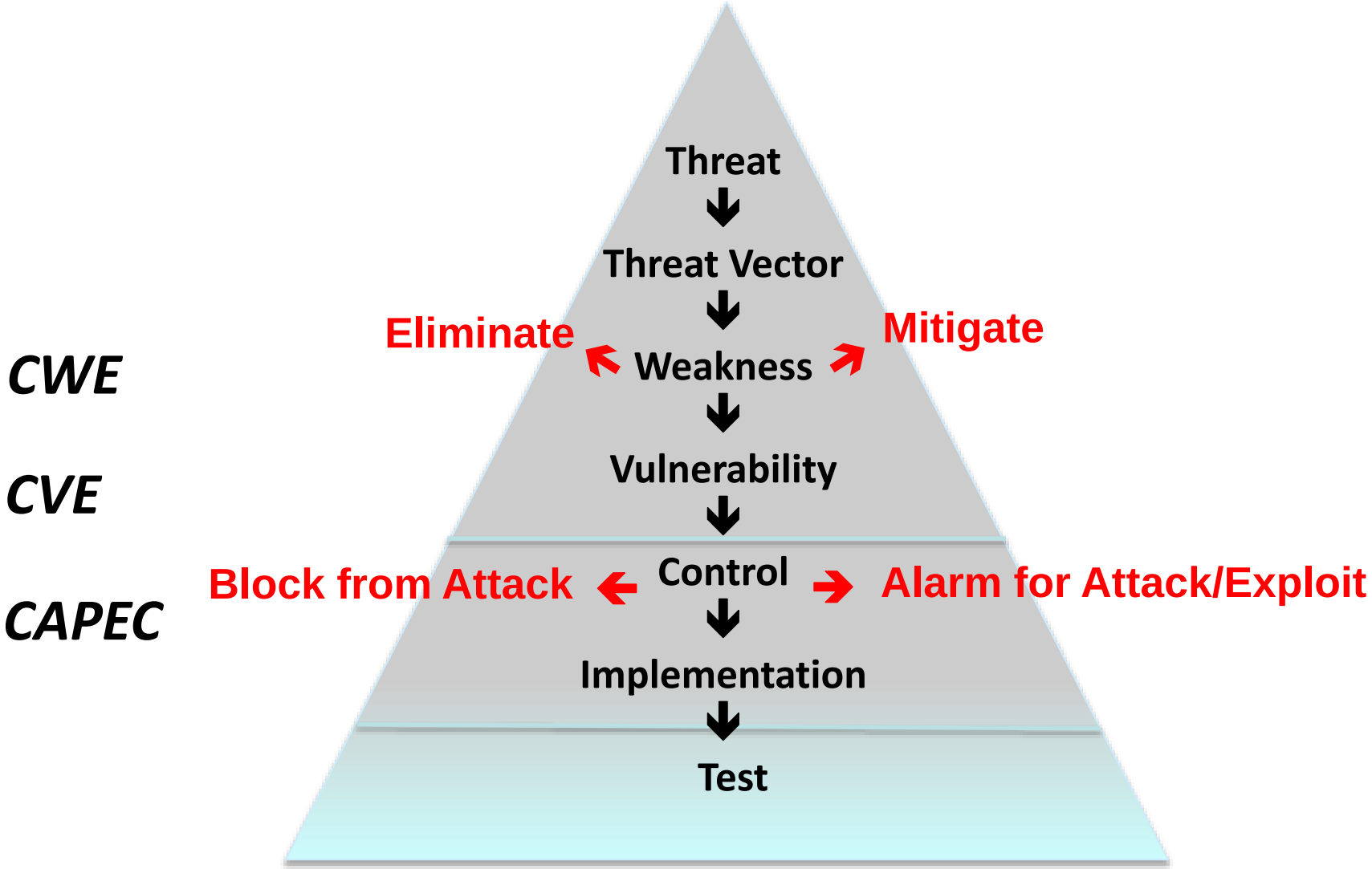
There are many examples of weaknesses:

A weakness is a development or manufacturing mistake, error, bug, flaw, defect, fault, anomaly, or the lack of control in software, firmware, hardware, supply chain process, or operational practice or behavior that could be *exploited.*


Attacks & Hazards
Mission Fulfillment
Weakness #1
Capability
Mission
Mission Fulfillment
Weakness #2
Weakness #3
Mission Fulfillment


Attacks & Hazards
Impact from Weakness #1
Mission Fulfillment
Exploitable Weakness #1 (a vulnerability)
Capability
Mission
Mission Fulfillment
Exploitable Weakness #2 (a vulnerability)
Weakness #3
Mission Fulfillment
Impact from Weakness #3

Assurance on the Management of Weaknesses



Leverage Common Weakness Enumeration (CWE) to mitigate risks to mission/business domains

CWE is a formal list of software weakness types created to:

- Serve as a common language for describing software security weaknesses in architecture, design, or code.
- Serve as a standard measuring stick for software security tools targeting these weaknesses.
- Provide a common baseline standard for weakness identification, mitigation, and prevention efforts.

Some Common Types of Software Weaknesses:

Buffer Overflows, Format Strings, Etc.
Structure and Validity Problems
Common Special Element Manipulations
Channel and Path Errors
Handler Errors
User Interface Errors
Pathname Traversal and Equivalence Errors
Authentication Errors
Resource Management Errors
Insufficient Verification of Data
Code Evaluation and Injection
Randomness and Predictability

***cwe**.mitre.org*

Prioritizing weaknesses to be mitigated

OWASP
The Open Web Application Security Project
http://www.owasp.org

OWASP Top 10

CWE/SANS Top 25

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

We would like a way to specify priorities based on business/mission risk

Common Weakness Risk Analysis Framework (CWRAF)

*How do I **identify** which of the 900+ CWE's are most important for my specific business domain, technologies and environment?*

Common Weakness Scoring System (CWSS)

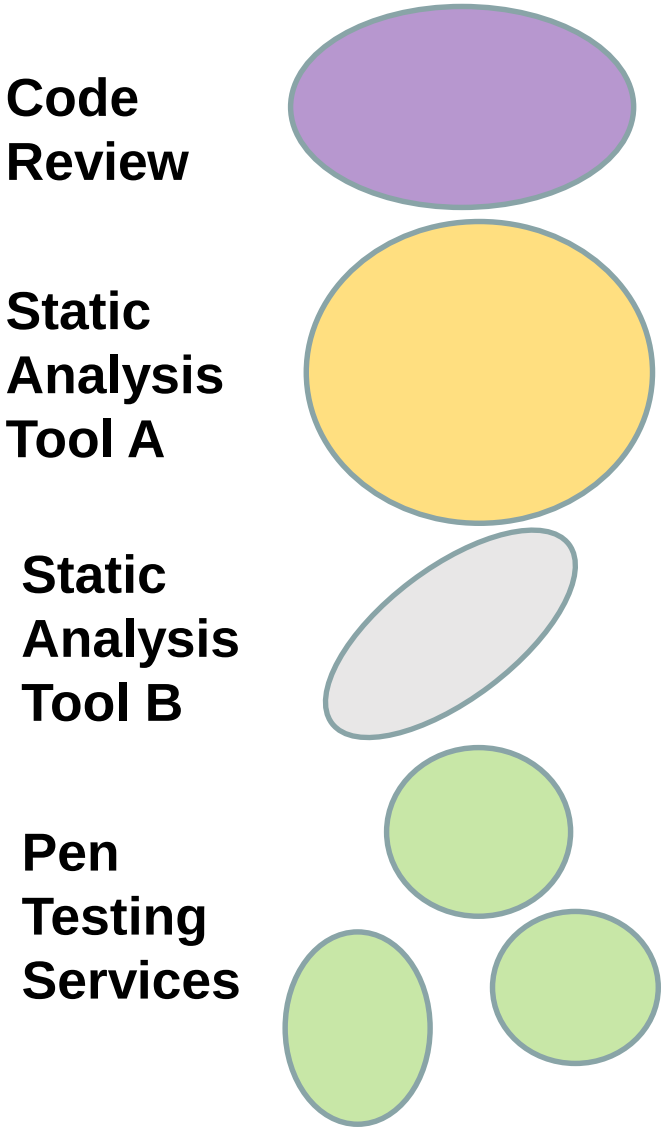
*How do I **rank** the CWE's I care about according to my specific business domain, technologies and environment?*

How do I identify and score weaknesses important to my organization?

CWE's all lead to these Technical Impacts

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

Utilizing a Priority List of Weaknesses



CWEs a capability *claims* to cover

Which static analysis tools and Pen Testing services find the CWEs I care about?

Take Advantage of the Multiple Detection Methods

- **Different assessment methods are effective at finding different types of weaknesses**
- **Some are good at finding the cause and some at finding the effect**

	Static Code Analysis	Penetration Test	Data Security Analysis	Code Review	*Architecture Risk Analysis*
Cross-Site Scripting (XSS)	X	X		X	
SQL Injection	X	X		X	
Insufficient Authorization Controls		X	X	X	*X*
Broken Authentication and Session Management		X	X	X	*X*
Information Leakage		X	X		*X*
Improper Error Handling	X				
Insecure Use of Cryptography		X		X	*X*
Cross Site Request Forgery (CSRF)		X		X	
Denial of Service	X	X	X		*X*
Poor Coding Practices	*X*			*X*	

Technical Impacts – Common Consequences

Detection Methods

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

TOP 25 MOST DANGEROUS SOFTWARE ERRORS | CWSS | CWRAF

Home > CWE List > CWE- Individual Dictionary Definition (2.5) | Search by ID: 78 Go

CWE List
Full Dictionary View
Development View
Research View
Reports
About
Sources
Process
Documents
FAQs
Community
Use & Citations
SwA On-Ramp
T-Shirt
Discussion List
Discussion Archives
Contact Us
Scoring
CWSS
CWRAF
CWE/SANS Top 25
Compatibility
Requirements
Coverage Claims Representation
Compatible Products
Make a Declaration
News
Calendar

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

Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

Applicable Platforms

Languages
All

Technology Classes
Database-Server

Modes of Introduction

This weakness typically appears in

Common Consequences

Scope	Effect
Confidentiality	Technical Impact: Read Since SQL databases SQL injection vulner
Access Control	Technical Impact: Bypa If poor SQL comman to a system as anoth
Access Control	Technical Impact: Bypa If authorization infor through the successf
Integrity	Technical Impact: Modi Just as it may be pos delete this informati

Likelihood of Exploit

Detection Methods

Automated Static Analysis

This weakness can often be detected using automated static analysis tools. Many modern tools use data flow analysis or constraint-based techniques to minimize the number of false positives.

Automated static analysis might not be able to recognize when proper input validation is being performed, leading to false positives - i.e., warnings that do not have any security consequences or do not require any code changes.

Automated static analysis might not be able to detect the usage of custom API functions or third-party libraries that indirectly invoke SQL commands, leading to false negatives - especially if the API/library code is not available for analysis.

This is not a perfect solution, since 100% accuracy and coverage are not feasible.

Automated Dynamic Analysis

This weakness can be detected using dynamic tools and techniques that interact with the software using large test suites with many diverse inputs, such as fuzz testing (fuzzing), robustness testing, and fault injection. The software's operation may slow down, but it should not become unstable, crash, or generate incorrect results.

Effectiveness: Moderate

Manual Analysis

Manual analysis can be useful for finding this weakness, but it might not achieve desired code coverage within limited time constraints. This becomes difficult for weaknesses that must be considered for all inputs, since the attack surface can be too large.

Demonstrative Examples

Example 1

Updated Detection Methods Launched 17 Feb 2014

CWE – Detection Methods

cwe.mitre.org/community/swa/detection_methods.html

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

Home > Community > Software Assurance > Detection Methods

Search by ID: Go

CWE List
Full Dictionary View
Development View
Research View
Reports
Mapping & Navigation
About
Sources
Process
Documents
FAQs
Community
Use & Citations
SwA On-Ramp
Discussion List
Discussion Archives
Contact Us
Scoring
Prioritization
CWSS
CWRAF
CWE/SANS Top 25
Compatibility
Requirements
Coverage Claims
Representation
Compatible Products
Make a Declaration
News
Calendar
Free Newsletter
Search the Site

Detection Methods

The "Detection Methods" field within many CWE entries conveys information about what types of assessment activities that weakness can be found by. Increasing numbers of CWE entries will have this field filled in over time. The recent Institute of Defense Analysis (IDA) State of the Art Research report conducted for DoD provides additional information for use across CWE in this area. Labels for the Detection Methods being used within CWE are:

- Automated Analysis
- Automated Dynamic Analysis
- Automated Static Analysis
- Black Box
- Fuzzing
- Manual Analysis
- Manual Dynamic Analysis
- Manual Static Analysis
- White Box

With this type of information (shown in the table below), we can see which of the specific CWEs that can lead to a specific type of technical impact are detectable by dynamic analysis, static analysis, and fuzzing evidence and which ones are not.

This table is incomplete, because many CWE entries do not have a detection method listed.

Technical Impact	Automated Analysis	Automated Dynamic Analysis	Automated Static Analysis	Black Box	Fuzzing	Manual Analysis	Manual Dynamic Analysis	Manual Static Analysis	White Box
Execute unauthorized code or commands		78, 120, 129, 131, 476, 805	78, 79, 98, 120, 129, 131, 134, 190, 426, 798, 805	79, 129, 134, 190, 426, 494, 698, 798		98, 120, 131, 190, 426, 494, 805	476, 798	78, 798	
Gain privileges / assume identity		601	306, 352, 426, 601, 798	259, 426, 798		259, 306, 352, 426	798	601, 798, 807	
Read data	209, 311, 327	78, 89, 129, 131, 209, 404, 665	78, 79, 89, 129, 131, 134, 352, 426, 798	14, 79, 129, 134, 319, 426, 798		89, 131, 209, 311, 327, 352, 426	209, 404, 665, 798	78, 798	14
Modify data	311, 327	78, 89, 129, 131	78, 89, 129, 131, 190, 352	129, 190, 319		89, 131, 190, 311,		78	

Section Contents
Software Assurance
Engineering for Attacks
Software Quality
Prioritizing Weaknesses
Detection Methods
Manageable Steps
Pocket Guides
Staying Informed
Finding More Information
Other Items of Interest
Discussion List
CWE Newsletter
Terms of Use

Technical Impact	Automated Analysis	Automated Dynamic Analysis	Automated Static Analysis	Black Box	Fuzzing	Manual Analysis	Manual Dynamic Analysis	Manual Static Analysis	White Box
Execute unauthorized code or commands		78, 120, 129, 131, 476, 805	78, 79, 98, 120, 129, 131, 134, 190, 426, 798, 805	79, 129, 134, 190, 426, 494, 698, 798		98, 120, 131, 190, 426, 494, 805	476, 798	78, 798	
Gain privileges / assume identity		601	306, 352, 426, 601, 798	259, 426, 798		259, 306, 352, 426	798	601, 798, 807	
Read data	209, 311, 327	78, 89, 129, 131, 209, 404, 665	78, 79, 89, 129, 131, 134, 352, 426, 798	14, 79, 129, 134, 319, 426, 798		89, 131, 209, 311, 327, 352, 426	209, 404, 665, 798	78, 798	14
Modify data	311, 327	78, 89, 129, 131	78, 89, 129, 131, 190, 352	129, 190, 319		89, 131, 190, 311, 327, 352		78	
DoS: unreliable execution		78, 120, 129, 131, 400, 476, 665, 805	78, 120, 129, 131, 190, 352, 400, 426, 805	129, 190, 426, 690	400	120, 131, 190, 352, 426, 805	476, 665	78	
DoS: resource consumption		120, 400, 404, 770, 805	120, 190, 400, 770, 805	190	400, 770	120, 190, 805	404	770	412
Bypass protection mechanism		89, 400, 601, 665	79, 89, 190, 352, 400, 601, 798	14, 79, 184, 190, 733, 798	400	89, 190, 352	665, 798	601, 798, 807	14, 733
Hide activities	327	78	78			327		78	

CWRAF/CWSS Provides Risk Prioritization for CWE throughout Software Life Cycle

- Enables education and training to provide specific practices for eliminating software fault patterns;
- Enables developers to mitigate top risks attributable to exploitable software;
- Enables testing organizations to use suite of test tools & methods (with CWE Coverage Claims Representation) that cover applicable concerns;
- Enables users and operation organizations to deploy and use software that is more resilient and secure;
- Enables procurement organizations to specify software security expectations through acquisition of software, hosted applications and services.

Security Automation "Pipework"

http://measurablesecurity.mitre.org/

Making Security Meas...

Making Security Measurable

A Collection of Information Security Community Standardization Activities and Initiatives

Home | About | MSM Directory | Incubator | Events & Participation | Search | Feedback

MITRE, in collaboration with government, industry, and academic stakeholders, is improving the measurability of security through **registries** of baseline security data, providing standardized **languages** as means for accurately communicating the information, defining proper **usage**, and helping establish community approaches for standardized **processes**.

Measurable security pertains at a minimum to the following areas:

- Software Assurance
- Application Security
- Asset Management
- Supply Chain Risk Management
- Cyber Intelligence Threat Analysis
- Cyber Threat Information Sharing
- Vulnerability Management
- Patch Management
- Configuration Management
- Malware Protection
- Intrusion Detection
- System Assessment
- Incident Coordination
- Enterprise Reporting
- Remediation

CVE – enabling reporting and patching of vulnerabilities

CWE – identifying and mitigating root cause exploitable weaknesses

CybOX – cyber observables and supply chain exploit indicators

CAPEC – schema attack patterns and software exploits

Getting Started in Software Assurance (SwA)

Success of the mission should be the focus of software and other assurance activities. Although increasing automation of various capabilities has provided great boons to our organizations, this automation is also at risk for becoming a targeted focus for attackers' attentions and techniques. Recognizing that your software and supply chain have exploitable weaknesses is a major step to improving the reliability, resilience, and integrity of your software when it faces attacks.

The key to gaining assurance about your software is to make incremental improvements when you develop it, when you buy it, and when others create it for you. No single remedy will absolve or mitigate all of the weaknesses in your software, or the risk. However, by blending several different methods, tools, and change in culture, one can obtain greater confidence that the important functions of the software will be there when they are needed and the worst types of failures and impacts can be avoided.

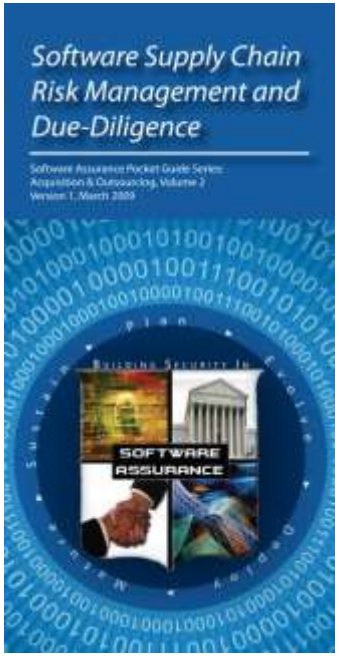
There is no crystal ball, or magic wand one can use to ensure software is *absolutely* secure against the unknown. However, there are ways to limit negative impacts and improve confidence in software-based capabilities and their ability to deliver their part to the organization's mission.

This section of the CWE Web site introduces specific steps you can take to 1) assess your individual software assurance situation and 2) compose a tailored plan to *strengthen* assurance of integrity, reliability, and resilience of your software and its supply chain. Learn more by following the links below:

- Engineering for Attacks
- Software Quality
- Prioritizing Weaknesses Based Upon Your Organization's Mission
- Detection Methods
- Manageable Steps
- Software Assurance Pocket Guide Series
- Staying Informed
- Finding More Information about Software Assurance

BACK TO TOP

Software Assurance (SwA) Pocket Guide Series



SwA in Acquisition & Outsourcing

- Software Assurance in Acquisition and Contract Language
- Software Supply Chain Risk Management and Due-Diligence

SwA in Development *

- Risk-based Software Security Testing
- Requirements and Analysis for Secure Software
- Architecture and Design Considerations for Secure Software
- Secure Coding and Software Construction
- Key Practices for Mitigating the Most Egregious Exploitable Software Weaknesses

* All include questions to ask developers

SwA Life Cycle Support

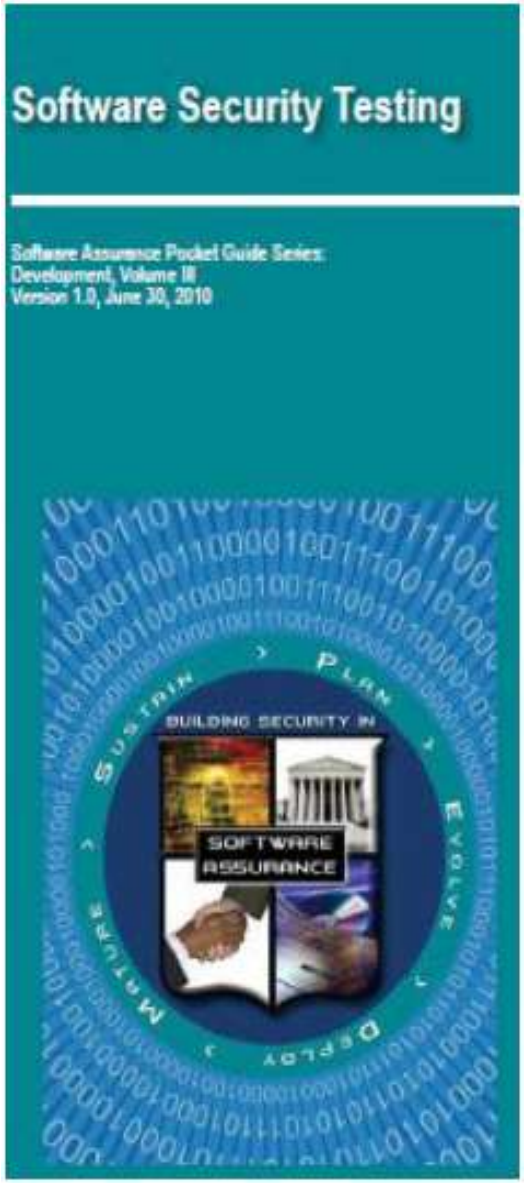
- SwA in Education, Training and Certification

SwA Pocket Guides and SwA-related documents are collaboratively developed with peer review; they are subject to update and are freely available for download via the DHS Software Assurance Community Resources and Information Clearinghouse at https://buildsecurityin.us-cert.gov/swa (see SwA Resources)

Homeland Security

Software Security Testing –vs– Security Requirements Testing

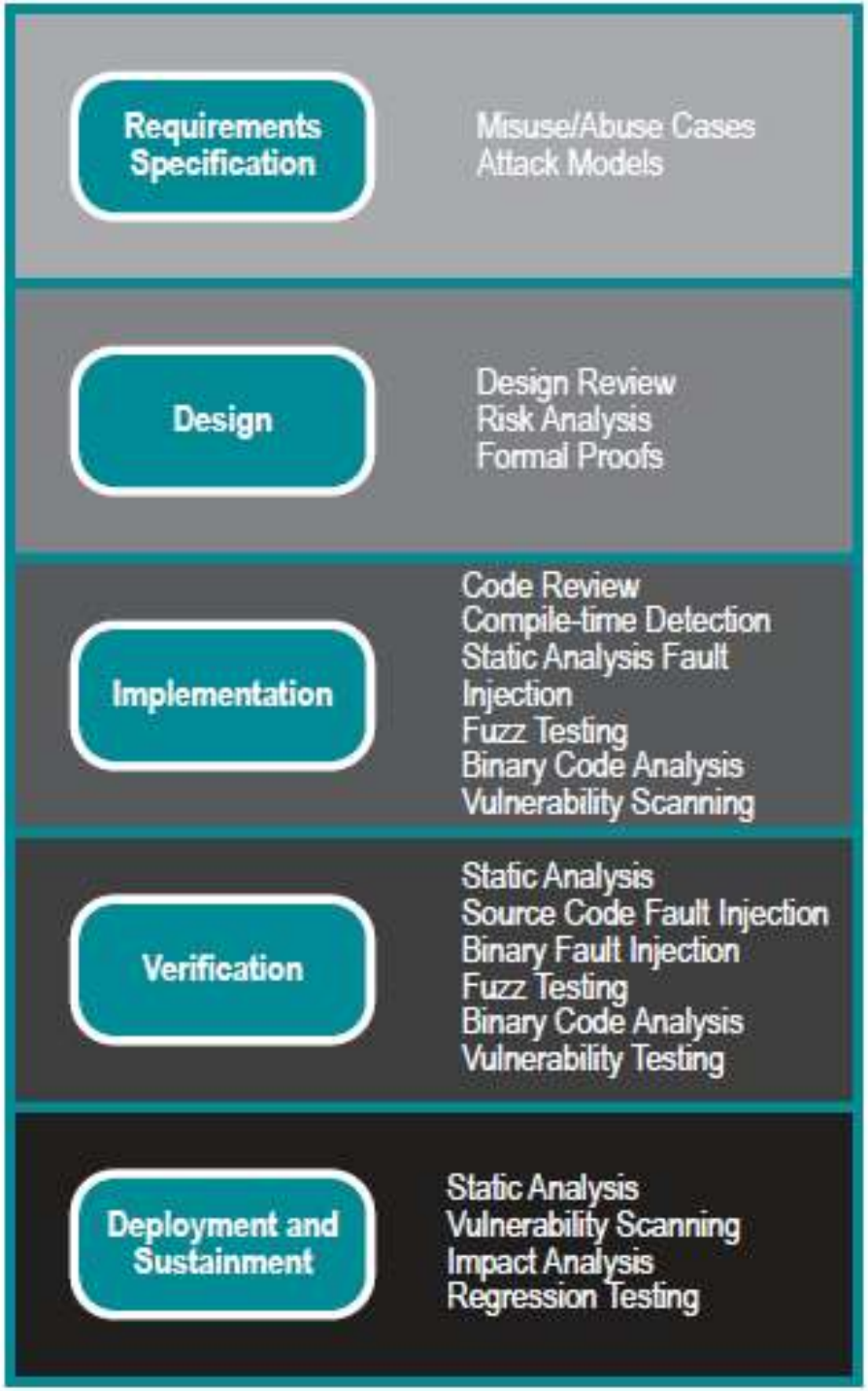
- Software security testing is not the same as testing the correctness and adequacy of security functions implemented by software, which are most often verified through requirements-based testing that:
  - cannot fully demonstrate that software is free from exploitable weaknesses / vulnerabilities.
  - is not the best approach to determining how software will behave under anomalous and hostile conditions.
- Download FREE SwA Pocket Guide on Software Security Testing at https://buildsecurityin.us-cert.gov/swa



Homeland Security

Software Security Test Techniques throughout SDLC

See details in FREE SwA Pocket Guide on Software Security Testing at https://buildsecurityin.us-cert.gov/swa



Penetration Testing can enhance pre-deployment test outcomes and identify post-release exploit points

Homeland Security

Key Practices for Mitigating the Most Egregious Exploitable Software Weaknesses

- *Identifies mission/business risks attributable to the respective weaknesses; identifies common attacks that exploit those weaknesses, and provides recommended practices for preventing the weaknesses.*
  - CWE focuses on stopping vulnerabilities at the source by educating designers, programmers, and QA/testers on how to eliminate all too-common mistakes before software is even shipped.
  - CWE Top-N lists serve as tools for education, training and awareness to help programmers prevent the kinds of vulnerabilities that plague the software industry.
  - Software consumers could use the same list to help them to ask for more secure software.
  - Software managers and CIOs can use the CWE list as a measuring stick of progress in their efforts to secure their software.

https://buildsecurityin.us-cert.gov/swa/downloads/KeyPracticesMWV13_02AM091111.pdf

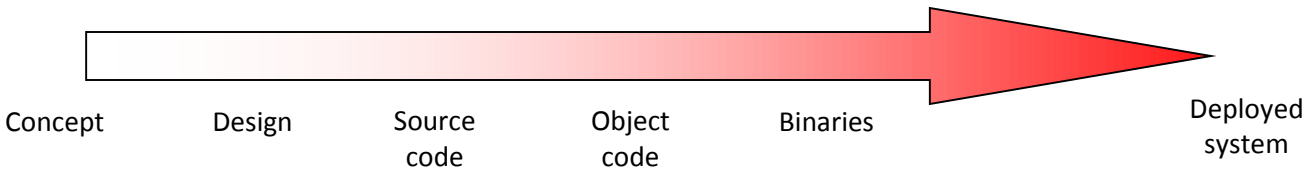
Key Practices: leveraging CWE & CAPEC

For a full listing and description of all the attacks related to a particular CWE visit the websites for CWE and CAPEC at http://cwe.mitre.org and http://capec.mitre.org .

Table 2 - CWEs and Their Related Attack Patterns and Mission/Business Risks

CWE	Related Attack Pattern	Mission/Business Risks
CWE-22 : Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	» CAPEC-23: File System Function Injection, Content Based » CAPEC-64: Using Slashes and URL Encoding Combined to Bypass Validation Logic » CAPEC-76: Manipulating Input to File System Calls » CAPEC-78: Using Escaped Slashes in Alternate Encoding » CAPEC-79: Using Slashes in Alternate Encoding » CAPEC-139: Relative Path Traversal	» DoS: crash / exit / restart » Execute unauthorized code or commands » Modify files or directories » Read files or directories
CWE-78 : Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	» CAPEC-6: TCP Header » CAPEC-15: Command Delimiters » CAPEC-43: Exploiting Multiple Input Interpretation Layers » CAPEC-88: OS Command Injection » CAPEC-108: Command Line Execution through SQL Injection	» DoS: crash / exit / restart » Execute unauthorized code or commands » Hide activities » Modify application data » Modify files or directories » Read application data » Read files or directories
CWE-79 : Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	» CAPEC-18: Embedding Scripts in Nonscript Elements » CAPEC-19: Embedding Scripts within Scripts » CAPEC-32: Embedding Scripts in HTTP Query Strings » CAPEC-63: Simple Script Injection » CAPEC-85: Client Network Footprinting (using AJAX/XSS) » CAPEC-86: Embedding Script (XSS) in HTTP Headers » CAPEC-91: XSS in IMG Tags » CAPEC-106: Cross Site Scripting through Log Files » CAPEC-198: Cross-Site Scripting in Error Pages » CAPEC-199: Cross-Site Scripting Using Alternate Syntax » CAPEC-209: Cross-Site Scripting Using MIME Type Mismatch » CAPEC-232: Exploitation of Privilege/Trust » CAPEC-243: Cross-Site Scripting in Attributes » CAPEC-244: Cross-Site Scripting via Encoded URI Schemes » CAPEC-245: Cross-Site Scripting Using Doubled Characters, e.g. %3C %3Cscript » CAPEC-246: Cross-Site Scripting Using Flash » CAPEC-247: Cross-Site Scripting with Masking through Invalid Characters in Identifiers	» Bypass protection mechanism » Execute unauthorized code or commands » Read application data

When should I focus on Weaknesses and Vulnerabilities?



Focus on Weaknesses

Focus on Vulnerabilities

A type of defect that ***may be*** exploitable.

Keep Weaknesses from becoming vulnerabilities

Something in code that ***can actually*** be exploited.

automation can help…

- Common Weakness Enumeration (**CWE**)
- Common Attack Pattern Enumeration and Classification (**CAPEC**)
- CWE Coverage Claims Representation (**CCR**)

- Common Weakness Enumeration (**CWE**)
- Common Weakness Risk Analysis Framework (**CWRAF**)
- Common Weakness Scoring System (**CWSS**)
- Common Attack Pattern Enumeration and Classification (**CAPEC**)
- CWE Coverage Claims Representation (**CCR**)

- Common Vulnerabilities and Exposures (**CVE**)
- Open Vulnerability Assessment Language (**OVAL**)
- Malware Attribute Enumeration and Characterization **(MAEC)**
- Cyber Obersvables eXpression **(CybOX)**

https://buildsecurityin.us-cert.gov/swa

Provides resources for stakeholders with interests in Software Assurance, Supply Chain Risk Management, and Security Automation

https://buildsecurityin.us-cert.gov

Focuses on making security a normal part of software engineering

The Rugged Software Manifesto

I am rugged... and more importantly, my code is rugged.

I recognize that software has become a foundation of our modern world.

I recognize the awesome responsibility that comes with this foundational role.

I recognize that my code will be used in ways I cannot anticipate, in ways it was not designed, and for longer than it was ever intended.

I recognize that my code will be attacked by talented and persistent adversaries who threaten our physical, economic, and national security.

I recognize these things - and I choose to be rugged.

I am rugged because I refuse to be a source of vulnerability or weakness.

I am rugged because I assure my code will support its mission.

I am rugged because my code can face these challenges and persist in spite of them.

I am rugged, not because it is easy, but because it is necessary... and I am up for the challenge.

Rugged?

ruggedsoftware.org

Software & Supply Chain Assurance:

Mitigating Risks Attributable to Exploitable ICT Products and Processes

Homeland Security

Joe Jarzombek, PMP, CSSLP
Director for Software & Supply Chain Assurance
Stakeholder Engagement & Cyber Infrastructure Resilience
Cyber Security & Communications

Enabling Enterprise Resilience through Security Automation, Software Assurance and Supply Chain Risk Management

Homeland Security