Estimating the Cost of Securing Software Applications

Brad Clark, PhD

18th Practical Software and Systems
Measurement Users' Group Meeting and Workshops
June 14, 2017

Abstract

- Making software applications secure from intrusion, corruption, attack, denial of service and other things is challenging. Does it really cost more to make software secure?
- This talk will discuss what it means to make software secure and where it might cost more to implement security measures.
- The COCOMO model is used to discuss costs associated with making software secure.

Topics

- Why Software Security
- Supply Chain Management Impact
- Examples of Software Weaknesses
- Software Component Security Requirements
- Software Development Security Requirements
- Follow-on Workshop Objectives

Why is Software Security Important?

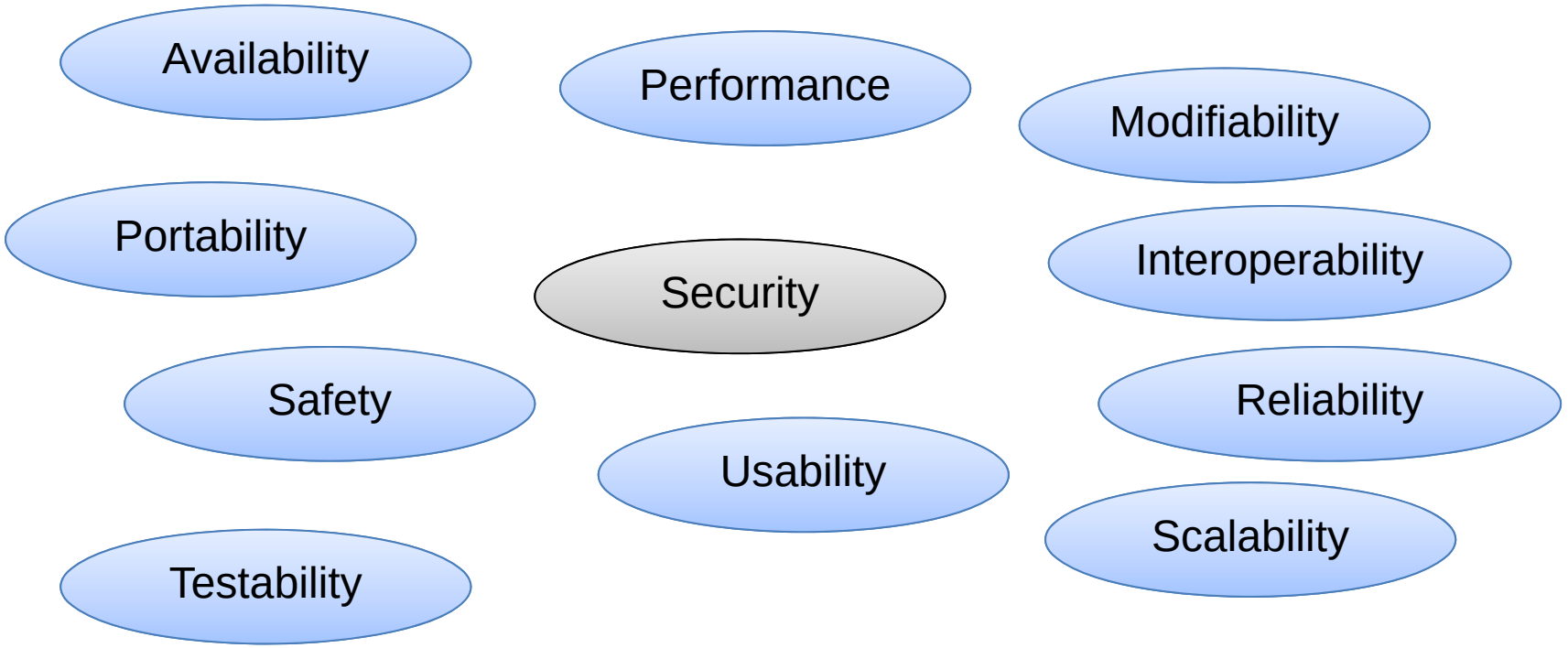
- There have been dramatic increases in business and mission risks attributable to exploitable software
- Software vulnerabilities jeopardize
  - intellectual property
  - consumer trust
  - business operations and services
  - broad spectrum of critical infrastructures (including everything from process control systems to commercial software products)
- Recent examples:
  - North Korean secret cyber unit 'likely behind' NSA ransomware attacks
  - Foreign hackers 'may have hit voter site days before referendum'
  - US child hacker launches cyber attack on Brussels Airport
  - Penthouse and Adult Friend Finder hack leaves over 412 million exposed… Oops

Is It Worth It?

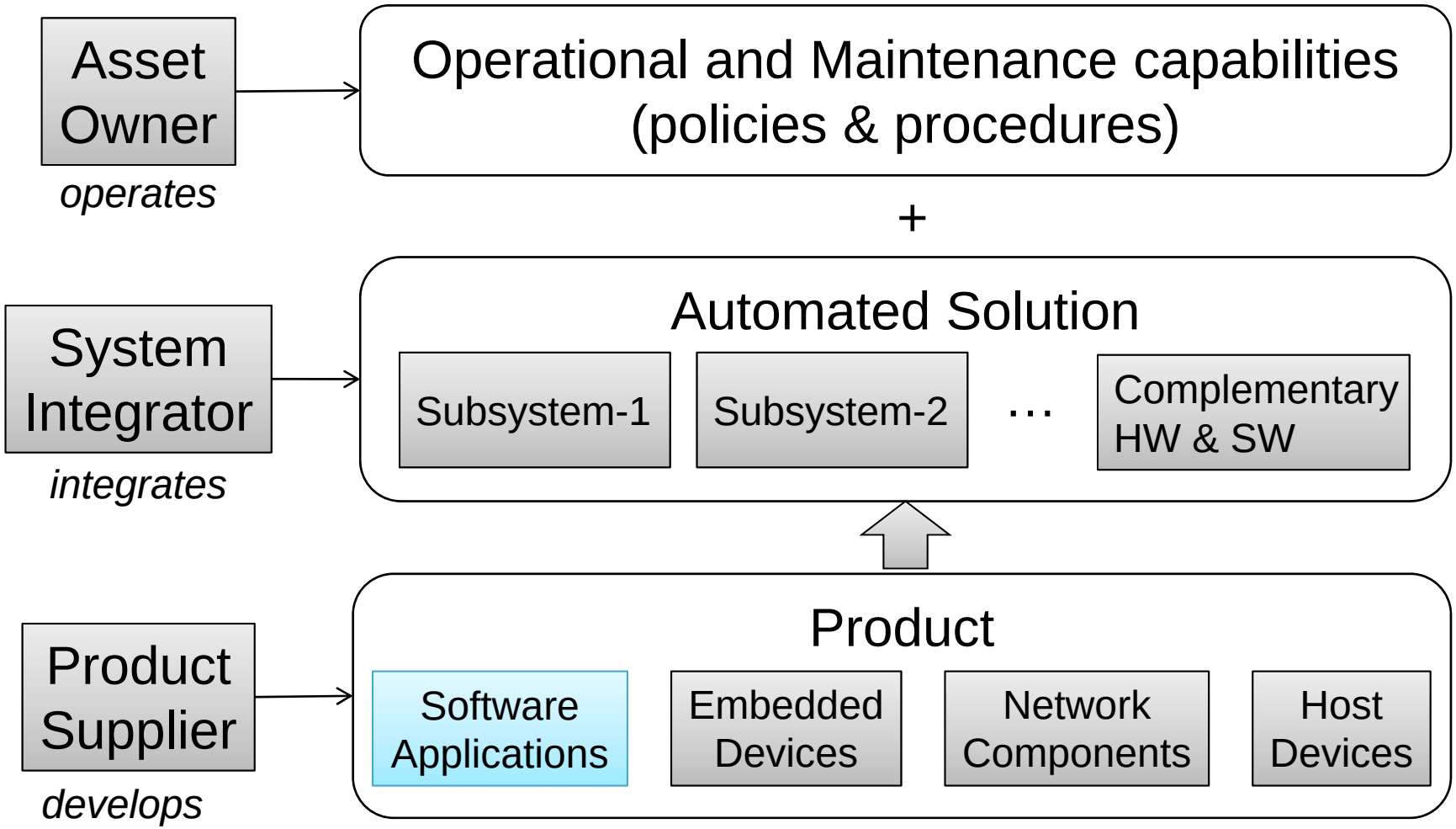
- How much additional effort (cost) does it take to develop secure software considering the impact of:
  - Security requirements for software
    - Levels of security
  - Implementation expertise
  - Testing independence
  - Process and tool support
  - Platform constraints and configurations (volatility)
- Two cost aspects:
  - Component security requirements
    - Software applications
    - Embedded devices
    - Host devices
    - Network components
  - Management of a secure development

Non-Functional Requirement Tensions

- Functional requirements specify the work for which the system is intended
- Non-Functional requirements pertain to the functions of the system
- There is a tradeoff between Security and other Non-Functional req’ts

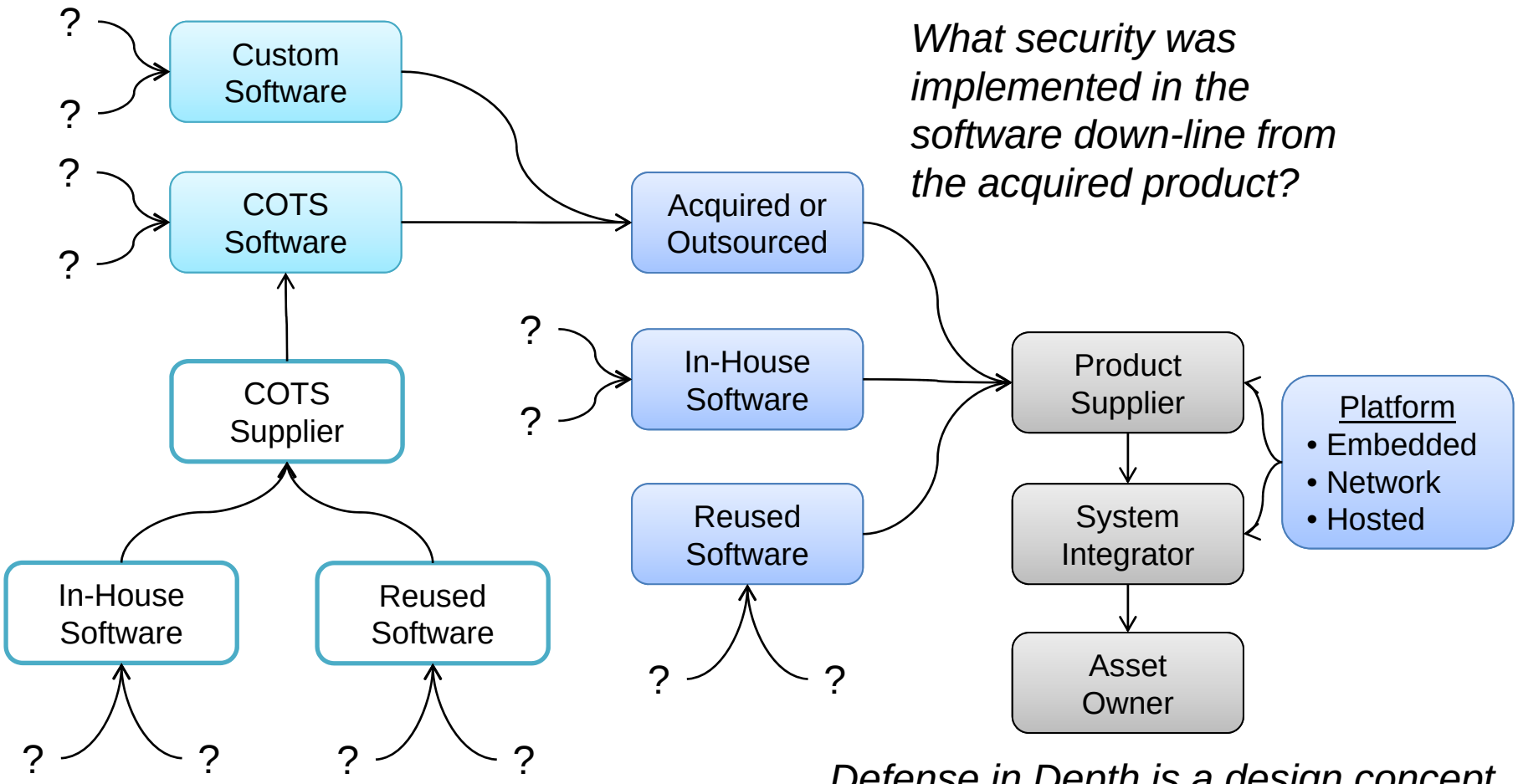


Application Development Context



Source: ISA-62443-4-1 Secure Product Development Lifecycle Requirements

Development Supply Chain Context



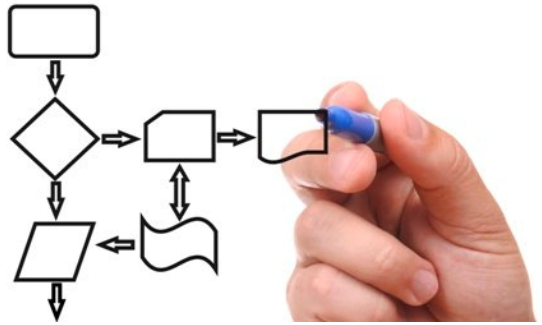
What security was implemented in the software down-line from the acquired product?

Defense in Depth is a design concept that attempts to address this issue.

Source: adapted from DHS, "Software Assurance in Acquisition and Contract Language"

Examples of Weaknesses Introduced During Design

- Acceptance of Extraneous Untrusted Data With Trusted Data
- Access to Critical Private Variable via Public Method
- Addition of Data Structure Sentinels, e.g. null character at the end of strings
- Algorithmic Complexity
- Allocation of File Descriptors or Handles Without Limits or Throttling
- Allocation of Resources Without Limits or Throttling
- Incorrect Control Flow Implementation
- Apple '.DS_Store'
- Argument Injection or Modification
- ASP.NET Misconfiguration: Not Using Input Validation Framework
- Asymmetric Resource Consumption (consume more resources than the access level permits)



Source: Mitre-CWE, Common Weakness Enumeration A Community-Developed List of Software Weakness Types

Weaknesses in the 2011 CWE/SANS Top 25 Most Dangerous Software Errors Examples

- Cross-Site Request Forgery (CSRF)
  - Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
  - Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
  - Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
- Porous Defenses
  - Execution with Unnecessary Privileges
  - Improper Restriction of Excessive Authentication Attempts
  - Incorrect Authorization
  - Incorrect Permission Assignment for Critical Resource
  - …

Examples of Weaknesses in SW Written in C++

- Access of Resource Using Incompatible Type ('Type Confusion')
- Access to Critical Private Variable via Public Method
- Base Addition of Data Structure Sentinel
- Assignment of a Fixed Address to a Pointer
- Buffer Access with Incorrect Length Value
- Base Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
- Buffer Underwrite ('Buffer Underflow')
- Cloneable Class Containing Sensitive Information
- Compiler Optimization Removal or Modification of Security-critical Code
  - …

Software Security Requirements

- What are examples of security requirements for software?
- Many security resources discuss security policy
- In this presentation, one set of requirements was selected to provide insight:
  - ISA-62443-4-2 Security for Industrial Automation And Control Systems Technical Security Requirements for IACS Components

  - ISA: International Society of Automation
  - IACS: Industrial automation and control system

Security Requirements for Software Components -1

1. Identification and authentication control

- Human user identification and authentication
- Software process and device identification and authentication
- Account management
- Identifier management
- Authentication management …

2. Use control

- Authorization enforcement
- Wireless control
- Use control for portable and mobile devices
- Session lock
- Remote session termination …

Source: ISA-62443-4-2 Technical Security Requirements for IACS Components

Security Requirements for Software Components -2

3. System integrity

- Communication integrity
- Malicious code protection
- Software and information integrity
- Input validation
- Error handling …

4. Data confidentially

- Information confidentiality
- Information persistence
- Use of cryptography

5. Restricted data flow

- Network segmentation
- Zone boundary protection
- Person-to-Person communication restrictions

Security Requirements for Software Components -3

6. Timely response to events

- Audit log accessibility
- Continuous monitoring

7. Resource availability

- Denial of service protection
- Resource management
- Control system backup, recovery and reconstitution
- Network and security configuration settings
- Least functionality

Component Security Levels

- The seven security requirements shown previously have four Security Levels (SL).
- Identify and authenticate all users (humans, software processes and devices) by mechanisms that

SL-1 – Protect against casual or coincidental access by unauthenticated entities.

SL-2 – Protect against intentional unauthenticated access by entities using simple means with low resources, generic skills and low motivation

SL-3 – Protect against intentional unauthenticated access by entities using sophisticated means with moderate resources

SL-4 - Protect against intentional unauthenticated access by entities using sophisticated means with extended resources

Source: ISA-62443-4-2 Technical Security Requirements for IACS Components

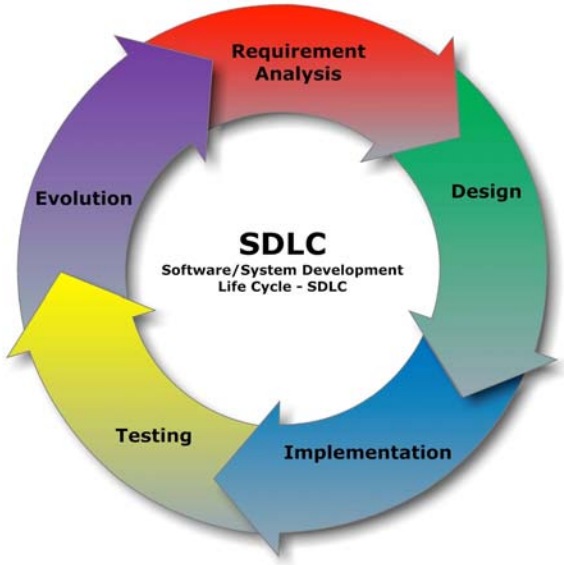
Impact of Component Security Requirements on Development Effort

- More requirements affect software effort (cost) by increasing the functionality (or size) to be implemented in the software
- The four security levels shown previously increase the amount of functionality (and size) and therefore effort
- The amount of effort required, directly related to the amount of functionality, is influenced by other factors such as
  - Product Factors (e.g. complexity, reliability)
  - Personnel Factors (capabilities, experience)
  - Platform Factors (e.g. constraints, volatility)
  - Project Factors (e.g. precedentedness, risk resolution, process capability, development flexibility, tools)

- These are addressed next

Secure Development Lifecycle -1

- Security management
  - Identification of responsibilities
  - Security expertise
  - Code signing
  - Development environment security
  - 3rd party embedded component security
  - Process verification



- Specification of security requirements
  - Product security requirements (authentication, authorization, encryption, auditing and other security capabilities)
  - Product security context (product's intended operating environment including physical environment)
  - Threat model (analysis that identifies potential security issues and how they will be addressed)
  - Security requirements review

Source: ISA-62443-4-1 Secure Product Development Lifecycle Requirements

Secure Development Lifecycle -2

- Secure by design
  - Secure design principles
  - Defense in depth design (layers of security)
  - Security design review
  - Assessing & addressing security-related issues
- Secure implementation
  - Security implementation review
  - Assessing & addressing security-related issues
- Security verification and validation testing
  - Security requirements testing
  - Threat mitigation testing
  - General vulnerability testing
  - Penetration testing

Secure Development Lifecycle -3

- Security defect management
  - Receiving notifications of security-related issues
  - Reviewing security-related issues
  - Assessing & addressing security-related issues
  - Disclosing security-related issues
- Security update management
  - Dependent component or operating system security update documentation
  - Security update delivery
  - Timely delivery of security patches

Conclusions

- Software component security requirements affect the amount of functionality
- Software development security requirements affect the productivity of the work

- Security Levels affect both the
  - Amount of functionality, e.g. more software to be developed
  - Amount of development tasks, e.g. increased reviews, testing, audits

COCOMO III Workshop

- Title: Implementing a New Driver for Software Security
- Facilitator: Brad Clark, USC Center for Systems and Software Engineering
- Prerequisites: An understanding of how a software cost estimation model is used in creating software development cost estimates. Knowledge of the COCOMO II Software Cost Estimation Model would be helpful but not absolutely necessary.
- Discussion: This workshop will begin with a brief overview of the COCOMO III project and the proposed cost estimation model.
  The main purpose of the workshop and the majority of time will be spent discussing how the cost for software component security requirements and development security requirements should be accommodated in the COCOMO III model.
- Goals/Products: A draft on how to estimate Required Software Security in COCOMO III and the associated driver's productivity range

Glossary

- CWE: Common Weakness Enumeration
- ISA: International Society of Automation
- IACS: Industrial Automation And Control System
- Vulnerability: A vulnerability is a software weakness that can be exploited by an attacker. Bugs and flaws collectively form the basis of most software vulnerabilities.
- Weakness: A weakness is an underlying condition or construct existing in a software system that has the potential for negatively impacting the security of the system.

Resources

- ISA-62443-4-1, “Secure Product Development Lifecycle Requirements,” Security for Industrial Automation and Control Systems, Draft 3, Edit 11, March 2016
- ISA-62443-4-2, “Technical Security Requirements for IACS Components,” Security for Industrial Automation and Control Systems, Draft 2, Edit 4, July 2, 2015
- Mitre-CWE, Common Weakness Enumeration A Community-Developed List of Software Weakness Types, http://cwe.mitre.org/data/index.html, accessed May 2017
- DHS, “Software Assurance in Acquisition and Contract Language,” Software Assurance Pocket Guide Series, Vol 1, Ver 1.2, May 2012