What Patterns Should We Use?

Carol Woody, Ph.D.
Robert Ellison, Ph.D.
Chuck Weinstock, Ph.D.

Software Engineering Institute (SEI)
Carnegie Mellon University (CMU)
Pittsburgh, PA 15213

Notices

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM21-0248

DARPA ARCOS Phase 1 Goal

"Risk of software deployment is acceptable"

Phase 1 Evaluation criteria: 3 of 6 evaluators agree with the generated evidence and confidence scores for 100,000+ nodes of curated data

Provide soundness and confidence score for the assurance case

How can patterns help?

- Pattern is a reusable argumentation framework
- Patterns exist at varying levels of abstraction
- Which ones provide value for the focus of the analysis?

Starting Thoughts…

Establishing what supports soundness and confidence

Principles can provide criteria for the analysis and support pattern selection:

- Intent: The defined intended behavior is correct and complete with respect to the desired behavior.
- Correctness: The implementation is correct with respect to its defined intended behavior, under foreseeable operating conditions.
- Innocuity: Any part of the implementation that is not required by the defined intended behavior has no unacceptable impact.

Reference: Holloway, C.M., *Understanding the Overarching Properties*, NASA/TM–2019–220292, Langley Research Center, 2019, https://ntrs.nasa.gov/search.jsp?R=20190029284 2020-05-07T13:15:54+00:00Z

Options to Explore

1 – Patterns in Code

2 – Patterns for Cyber Attack

3 – Patterns of Standards

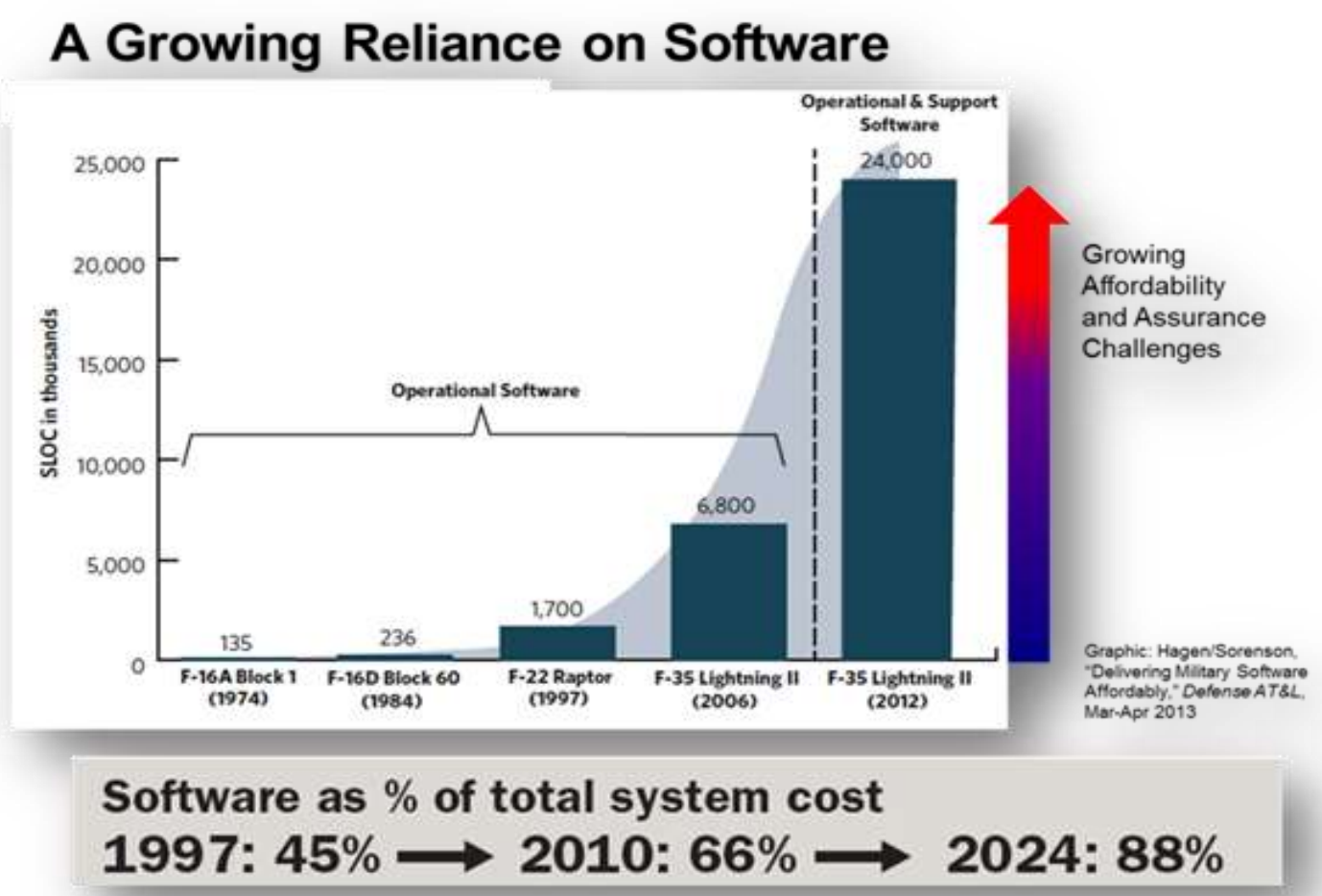
4 – Patterns of Actual Practice

5 – Patterns of Risk

6 – Patterns of Archetypes

Patterns in Code

Patterns of Software Usage



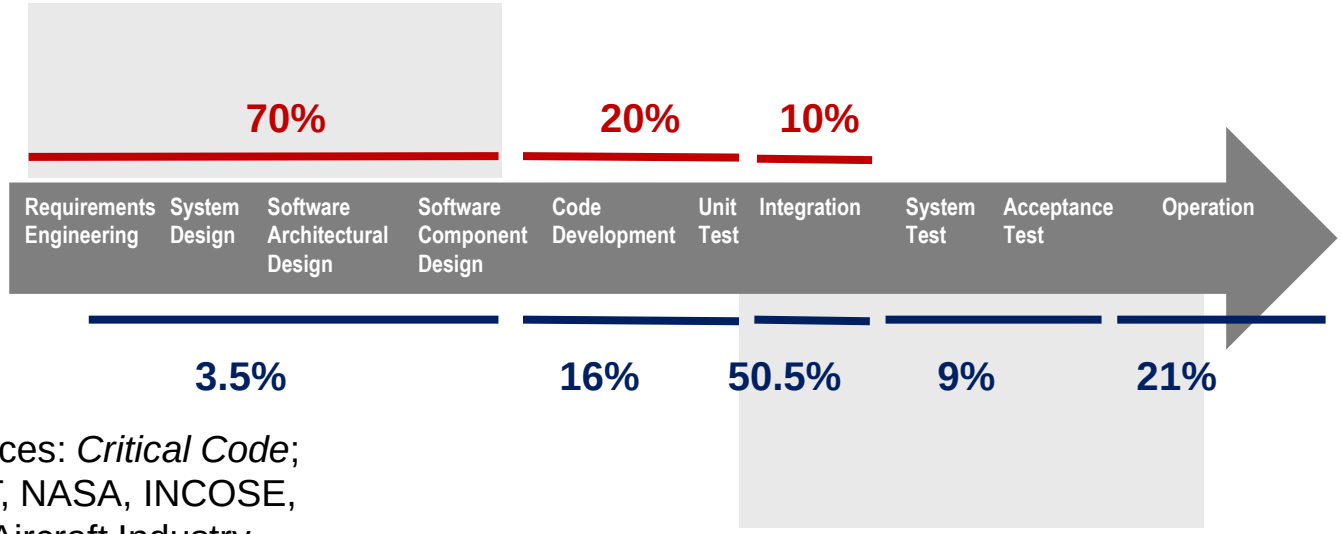
Does not include operating software on standard commodities such as laptops, iPads, etc.

Windows 10 has some 50 million lines of code

Source: U.S. Air Force Scientific Advisory Board. *Sustaining Air Force Aging Aircraft into the 21st Century* (SAB-TR-11-01). U.S. Air Force, 2011.

Patterns of Software Defects

Where Software Defects Are Introduced



Sources: *Critical Code*; NIST, NASA, INCOSE, and Aircraft Industry Studies)

Where Software Defects Are Found

All software has defects:

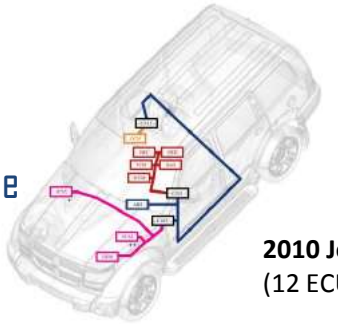
Best-in-class code has <600 defects per million lines of code (MLOC).
Good code has around 1000 defects per MLOC.
Average code has around 6000 defects per MLOC.

(based on Capers Jones research http://www.namcook.com/Working-srm-Examples.html)

Patterns of Reuse

Assemble from 3rd party components (COTS, GOTS, Open Source) to reduce construction cost/schedule and increase flexibility

Example:
Vehicles are now Assembled from Engine Control Units (ECUs)



2010 Jeep Cherokee
(12 ECUs)

2014 Jeep Cherokee
(32 ECUs)

Dept of Commerce has proposed the Software Bill of Materials (SBOM)

ECUs are prefabricated, software-driven components addressing select functionality and tailorable to a specific domain.

Modern high-end automotive vehicles have software and connectivity:

- Over 100 million lines of code
- Over 50 antennas
- Over 100 ECUs

Sources: Miller and Valasek, A Survey of Remote Automotive Attack Surfaces, http://illmatics.com/remote%20attack%20surfaces.pdf; https://www.cst.com/webinar14-10-23~?utm_source=rfg&utm_medium=web&utm_content=mobile&utm_campaign=2014series
https://en.wikipedia.org/wiki/Electronic_control_unit

Adding Soundness and Confidence?

Applying the Principles:

- Intent
  - mapping of requirements to tests
  - mapping functionality to code
- Correctness
  - reduced defects in developed code
  - enhanced information about 3rd party code which is often limited
- Innocuity
  - code not related to safety and security does not corrupt either

Patterns for Cyber Attack in Code

The Attacker Needs Three Ingredients

Exploitable vulnerabilities

- Millions of lines of software code contain defects; up to 5% are potential vulnerabilities

  ref: Woody, Carol et al. *Predicting Software Assurance Using Quality and Reliability Measures.* http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=428589)

- Hundreds of thousands of known software vulnerabilities exist

  ref: NIST National Vulnerability Database, https://nvd.nist.gov/general/nvd-dashboard.

Access

- Increased connectivity links systems to other systems and connects new types of devices (IoT), which may be inadequately protected.
- Increased system and device connectivity with trusted connections provide security gaps that may be compromised.

Ability to exploit

- Attackers have access to software development tools and techniques as well as libraries of successful exploit software
- Attackers can apply reverse engineering to commercial and open source software to discover weaknesses.

Collections of Vulnerability Tracking Patterns

Common Vulnerability Enumeration

(http://cve.mitre.org/index.html)

Library of known vulnerabilities tagged by a CVE ID recognized internationally for reporting vulnerability data from many sources

Common Weakness Enumeration

(http://cwe.mitre.org/index.html)

Community-developed list of common software security weaknesses that can lead to software vulnerabilities

Top 25 were identified to focus attention on the most common problems

National Vulnerability Database

(https://nvd.nist.gov)

U.S. government repository of standards based vulnerability management data represented in a format that enables automation of vulnerability management, security measurement, and compliance

Patterns of Code Attacks

Common Attack Pattern Enumeration and Classification CAPEC™

Provides a comprehensive dictionary of known patterns of attack employed by adversaries to exploit known weaknesses in cyber-enabled capabilities. It can be used by analysts, developers, testers, and educators to advance community understanding and enhance defenses.

[Table of types of attacks provided on the following slide]

CAPEC Meta Abstractions			
Exploitation of Trusted Identifiers (21)	Buffer Manipulation (123)	Code Inclusion (175)	Information Elicitation (410)
Exploiting Trust in Client (22)	Shared Data Manipulation (124)	Configuration/Environment Manipulation (176)	Manipulate Human Behavior (416)
Forced Deadlock (25)	Flooding (125)	Software Integrity Attack (184)	Modification During Manufacture (438)
Leveraging Race Conditions (26)	Pointer Manipulation (129)	Reverse Engineering (188)	Manipulation During Distribution (439)
Fuzzing (28)	Excessive Allocation (130)	Protocol Analysis (192)	Hardware Integrity Attack (440)
Manipulating User State (74)	Resource Leak Exposure (131)	Functionality Misuse (212)	Malicious Logic Insertion (441)
Man in the Middle Attack (94)	Parameter Injection (137)	Communication Channel Manipulation (216)	Physical Theft (507)
Brute Force (112)	Content Spoofing (148)	Fingerprinting (224)	Contaminate Resource (548)
API Manipulation (113)	Identity Spoofing (151)	Sustained Client Engagement (227)	Local Execution of Code (549)
Authentication Abuse (114)	Input Data Manipulation (153)	Privilege Escalation (233)	Functionality Bypass (554)
Authentication Bypass (115)	Resource Location Spoofing (154)	Resource Injection (240)	Object Injection (586)
Excavation (116)	Infrastructure Manipulation (161)	Code Injection (242)	Traffic Injection (594)
Interception (117)	File Manipulation (165)	Command Injection (248)	Obstruction (607)
Privilege Abuse (122)	Footprinting (169)	Protocol Manipulation (272)	Fault Injection (624)
	Action Spoofing (173)	Bypassing Physical Security (390)	

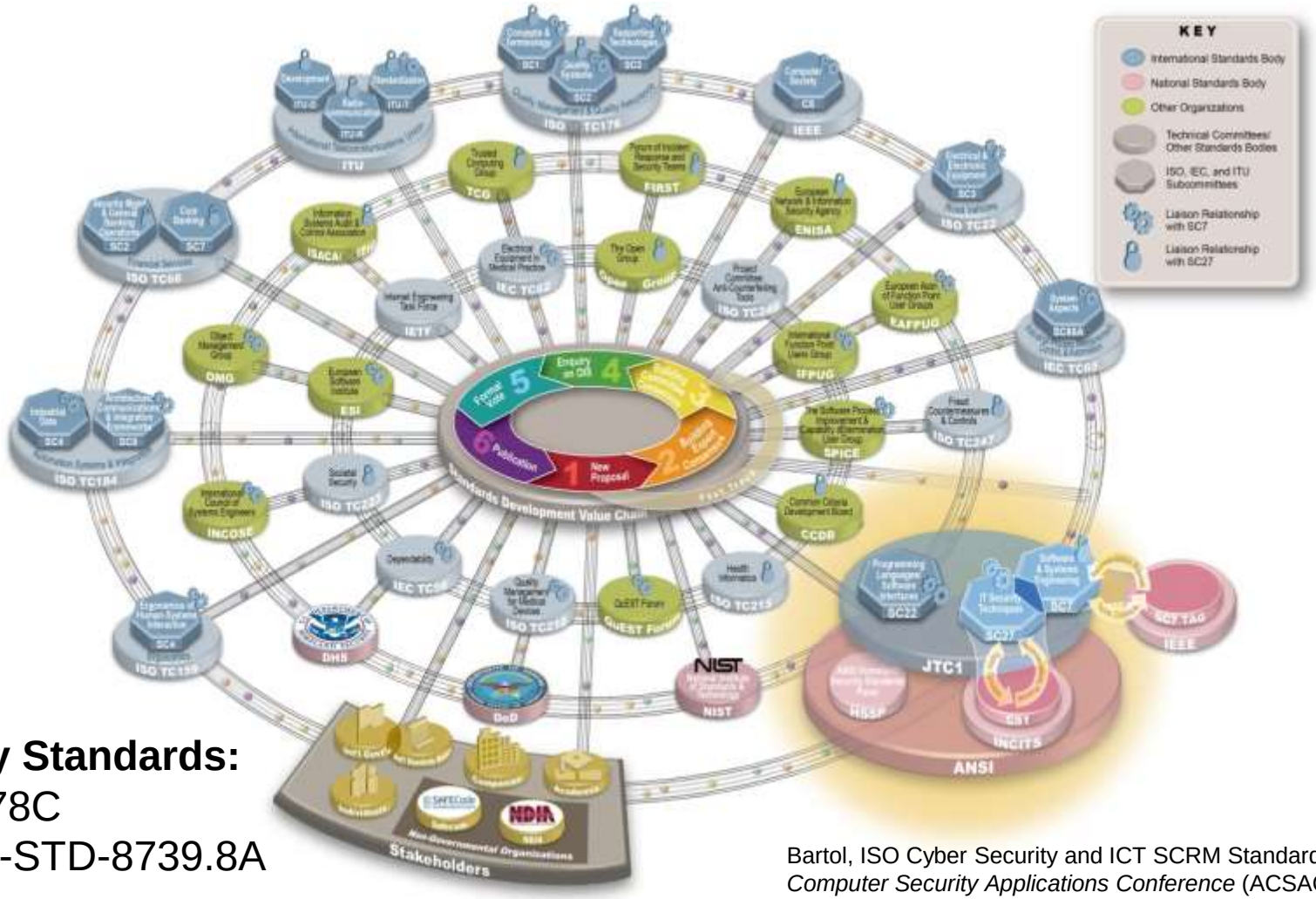
Adding Soundness and Confidence?

Applying the Principles

- Intent
  - Tracking of reduced vulnerabilities to reduce attacker impairing results
  - Reduction of attack surface to limit attacker access
- Correctness
  - Engineering practices that support removal of unacceptable known vulnerabilities and weaknesses
- Innocuity
  - Pruning functionality to reduce attack surface to minimize potential harm

Patterns of Standards & Certifications

Too Many Software and Cybersecurity Standards



Safety Standards:
DO-178C
NASA-STD-8739.8A

Bartol, ISO Cyber Security and ICT SCRM Standards, *Annual Computer Security Applications Conference* (ACSAC), December 2012 www.acsac.org/2010/program/case/wed-1330-Bartol.pdf

Acquisition Certification Programs

DoD: Cybersecurity Maturity Model Certification (CMMC) – assessing a vendor's implementation of cybersecurity practices and the company's maturity processes (DFARS Case 2019-D041 effective November 30, 2020)

GSA: Federal Risk and Authorization Management Program (FedRAMP) is a government-wide program that provides a standardized approach to security assessment, authorization, and continuous monitoring for cloud products and services

Department of Defense (DoD), GSA, and the National Aeronautics and Space Administration (NASA) jointly issue the Federal Acquisition Regulation (FAR) for use by executive agencies in acquiring goods and services.

Adding Soundness and Confidence?

Standards define broad ranges of practices that should be used, but without tailoring may have no direct baring on the actual system.

Certifications can be thought of as desired behaviors.

Applying the Principles

- Intent
  - Standards that define appropriate safety and security constraints are applied to the system
- Correctness
  - Selected standards and certifications have been met (requirements)
- Innocuity
  - Parts of the system not constrained by the standards and certification do not impact safety and security

Patterns of Actual Practices

What are Successful Organizations Doing

The Building Security In Maturity Model (BSIMM) is a multi-year study (since 2006) led by Cigital and Fortify (http://bsimm.com/)

Reviewed the efforts of **over 130 organizations**

- Nine verticals: Financial services, FinTech, Independent software vendors, Technology, Retail, Cloud, Healthcare, Internet of Things, and Insurance.
- Latest version, BSIMM-11, was just released.

Objective was to identify what is currently done

- Establishes state of the practice
- Mechanism to compare yourself to industry peers

Building Security In Maturity Model (BSIMM)

121 security activities across the surveyed organizations that represent 12 major practice areas.

Three levels of levels of maturity for each practice based on usage.

- Level 1 activities (straightforward and simple) are commonly observed
- Level 2 (more difficult and requiring more coordination) slightly less so
- Level 3 (rocket science) are much more rarely observed.

All agree that the success of their initiative hinges on having an internal group devoted to software security.

BSIMM Average Scores

Governance

Intelligence

Development

Deployment

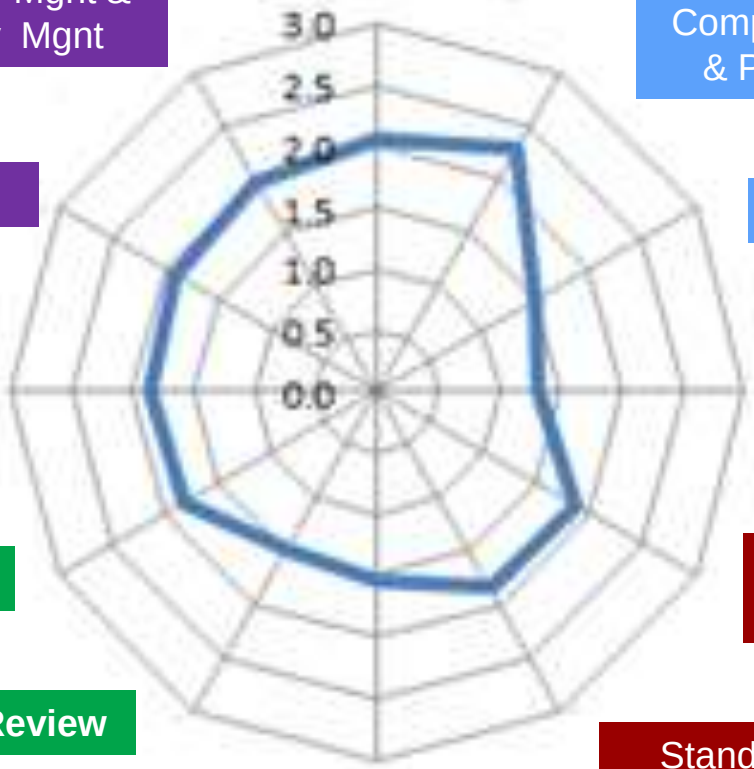
Strategy & Metrics

Configuration Mgnt & Vulnerability Mgnt

Compliance & Policy

Software Environment

Training



Penetration Testing

Attack Models

Security Testing

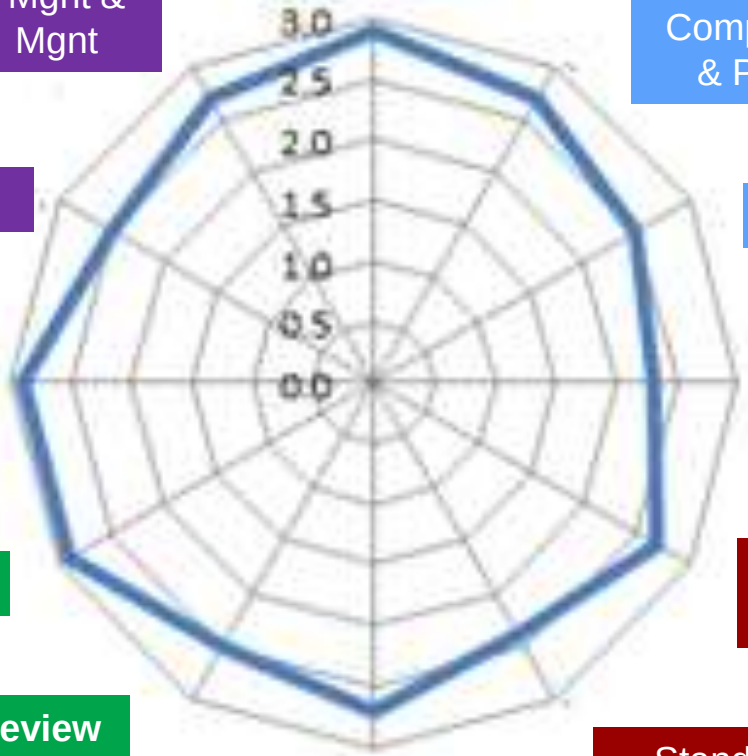
Security Features and Designs

Code Review

Standards & Requirements

Architecture Analysis

BSIMM Scores for Top 10 Firms



Governance

Intelligence

Development

Deployment

Strategy & Metrics

Configuration Mgnt & Vulnerability Mgnt

Compliance & Policy

Software Environment

Training

Penetration Testing

Attack Models

Security Testing

Security Features and Designs

Code Review

Standards & Requirements

Architecture Analysis

Adding Soundness and Confidence?

BSIMM defines practices that are actually used by an organization but are not connected to a specific system. This is a self-assessment.

Since this evaluation is at the organizational level, it is not clear how to apply principles for a specific system.

Applying the Principles

- Intent?
- Correctness?
- Innocuity?

Patterns of Risk

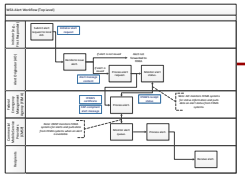
Mission Context for Security Risk Scenarios

Outcomes

- Data disclosure (Confidentiality)
- Data modification (Integrity)
- Insertion of false data (Integrity)
- Destruction of data (Availability)
- Interruption of access to data (Availability)

Affects

Workflow / Mission Thread

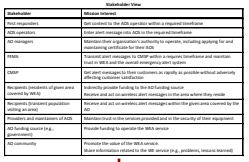


Produces

Adverse Mission Consequences / Losses

Affects

Stakeholder Interests

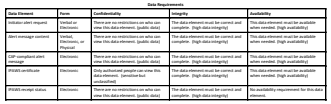


Produces

Adverse Stakeholder Consequences / Losses

Produces

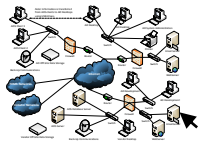
Mission Data



Targets

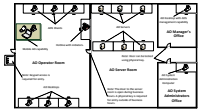
Technology Environment

Technology Infrastructure



Use Cases

Facilities



Entity of Interest

Exploits weaknesses and vulnerabilities

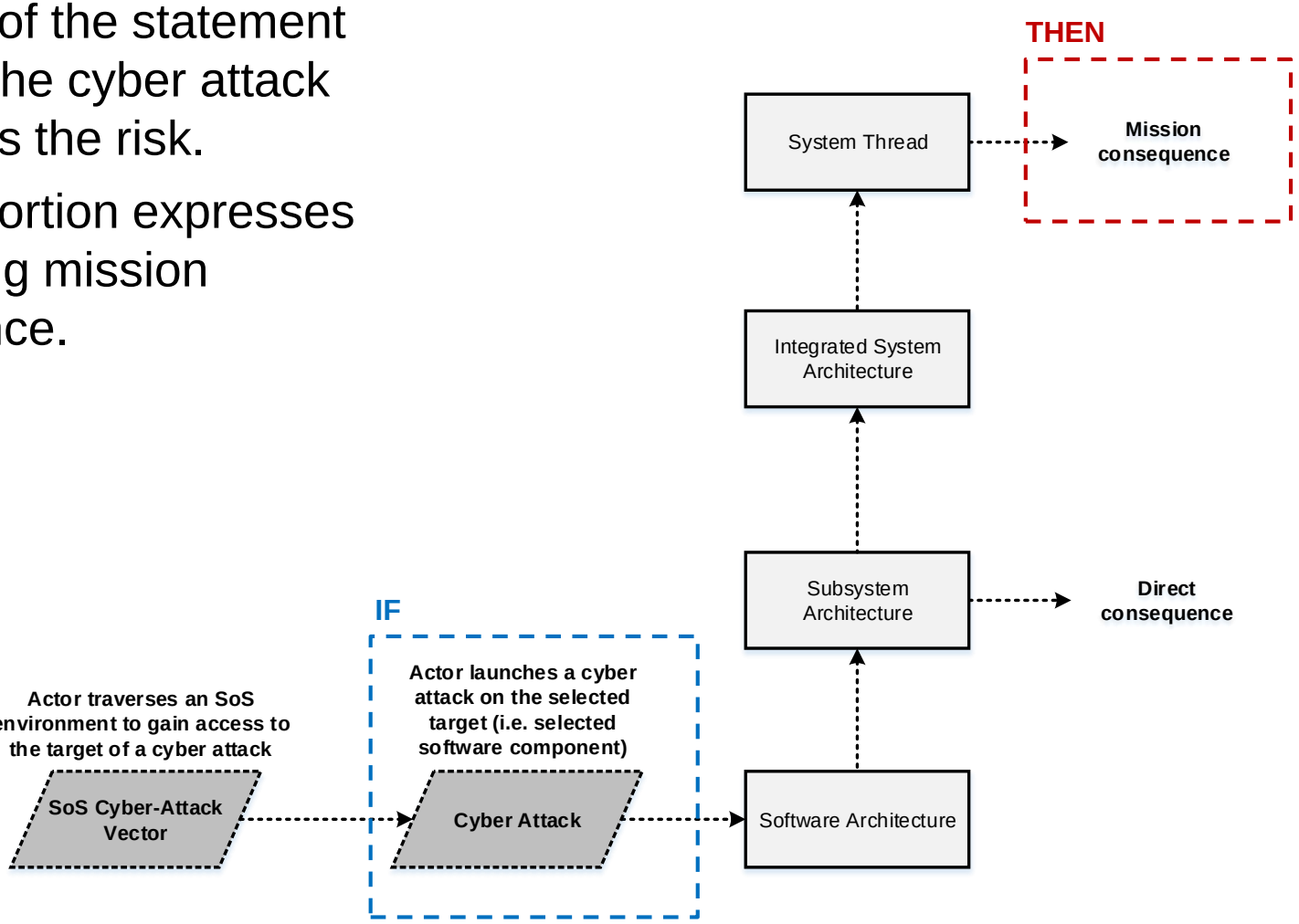
Threat Actor

We develop security risk scenarios to analyze the mission impact of data security breaches.

Example Risk Statement

Risk statements describe the mission risk if the threat is realized

- The *if* part of the statement describes the cyber attack that triggers the risk.
- The *then* portion expresses the resulting mission consequence.



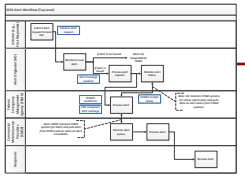
Selecting Controls to Address Risk

Outcomes

- Data disclosure (Confidentiality)
- Data modification (Integrity)
- Insertion of false data (Integrity)
- Destruction of data (Availability)
- Interruption of access to data (Availability)

Affects

Workflow / Mission Thread

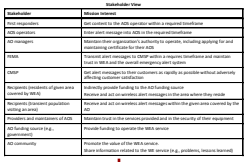


Produces

Adverse Mission Consequences / Losses

Affects

Stakeholder Interests

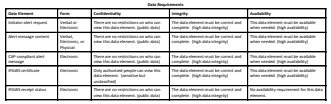


Produces

Adverse Stakeholder Consequences / Losses

Produces

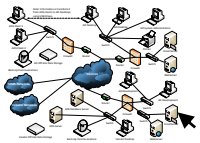
Mission Data



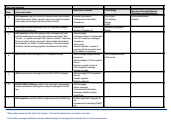
Targets

Technology Environment

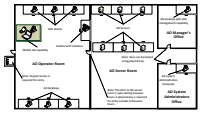
Technology Infrastructure



Use Cases



Facilities



Entity of Interest

Threat Actor

Exploits weaknesses and vulnerabilities

Reduce mission consequence amplifiers

Reduce stakeholder consequence amplifiers

Controls

Reduce threat enablers (e.g., weaknesses, vulnerabilities)

Adding Soundness and Confidence?

Applying the Principles

- Intent
  - the system is design addresses scenarios of expected threat and abuse
- Correctness
  - verification that planned controls are in place for known threats
- Innocuity
  - parts of the system not impacted by threats and controls do not increase safety and security risk

Patterns of Risk: Archetypes

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Archetype

A generalized pattern that illustrates the key characteristics of a complex threat scenario

- An access archetype defines how an actor can gain access to the target of an attack (i.e., defines an access path).
- Examples:
  - An actor in the supply chain implants malicious code in a software update.
  - An insider uses network access to attack a system.
  - An insider uses physical access to launch an attack.
  - An actor exploits trusted third-party access to an organization's systems and networks.
  - An actor spoofs the identify of a legitimate user.
  - An external attacker breaches perimeter security controls.
  - Others

Key Elements of Security Risk Scenarios

Threat

- Access path
  - Description of threat actor
  - Identification of the system or component being targeted
  - Description of how the actor will gain access to the target of the attack
- Cyber attack
  - Description of the attack on the targeted system or component
  - Description of the outcome (i.e., direct consequence on data assets)

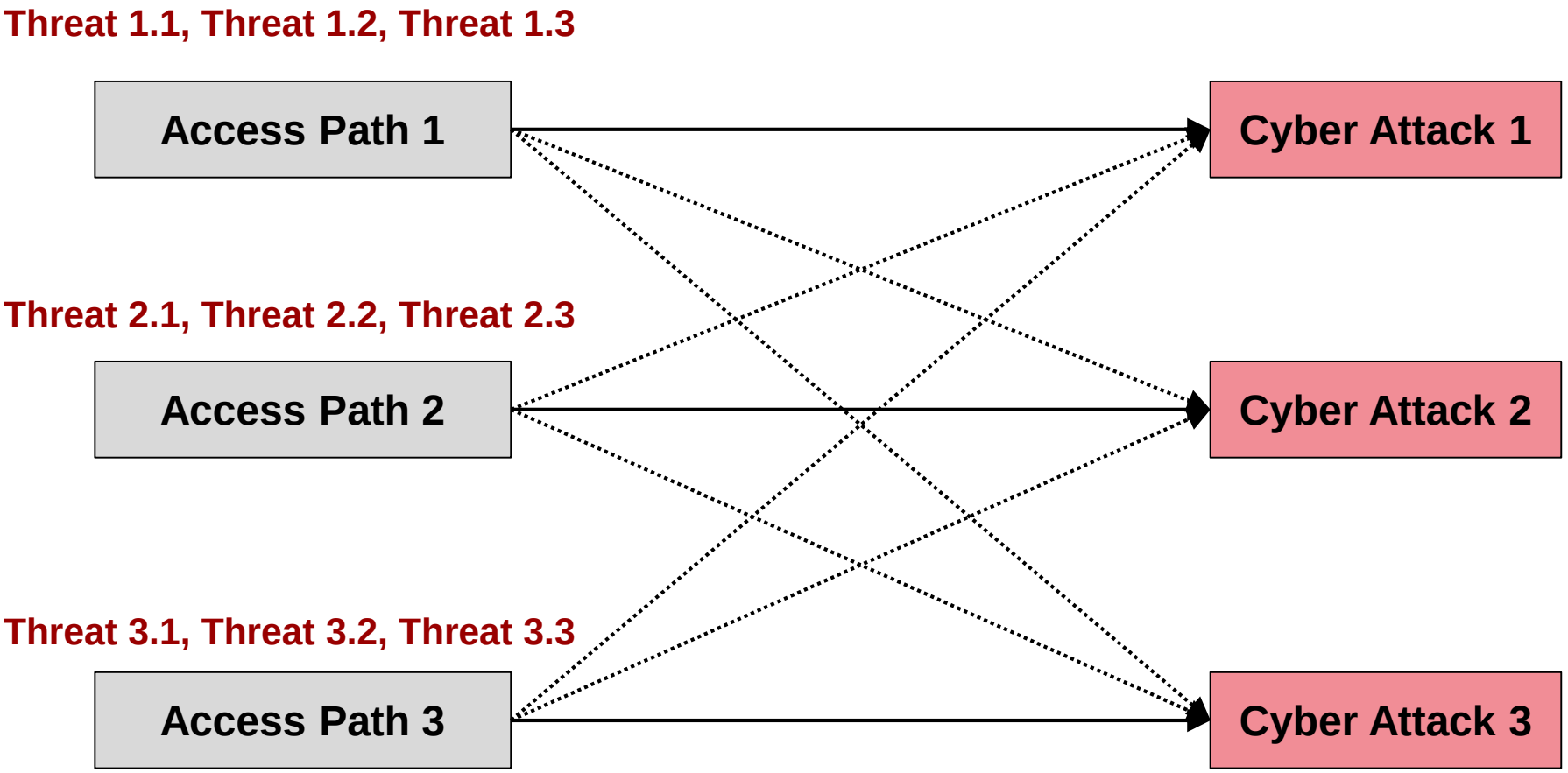
Consequences

- Mission thread consequences
- Stakeholder consequences

Threat Identification: *Access to Attack Mapping*

A threat is the connection of an access path to a cyber attack

Threat Identification: *Many to Many Mappings*



Adding Soundness and Confidence?

Applying the Principles

- Intent
  - comparing archetypes to requirements would show potential completeness and gaps in consideration of potential threats
- Correctness
  - archetypes can be composed of CAPEC attack patterns; connecting consideration of attacks in requirements and verification through testing
- Innocuity
  - parts of the system not addressed by the archetypes do not increase safety and security risk

Summary

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Existing Patterns all Have Limitations

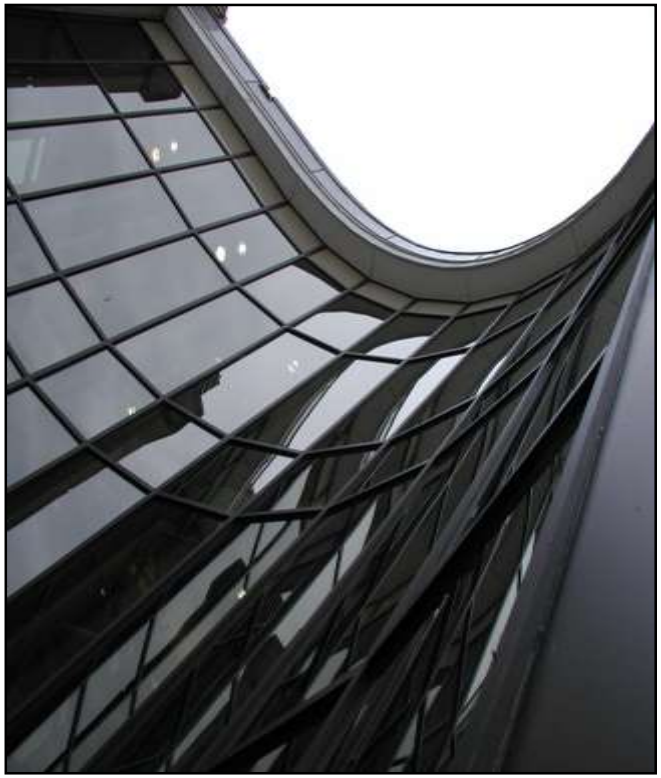
Where should we focus?

- Mission scenarios (patterns of risk) and archetypes seem to provide the greatest value for assurance
  - Archetypes can capture patterns of mitigations that improve assurance
- Code level patterns may not reflect the composition at the system level
  - CWEs would be patterns of expected code mitigations
- Organizational level patterns may not reflect what was actually done for the system under analysis

Additional Observations

- Recognize that the artifacts used in the ARCOS analysis will resemble patterns of practice (self-assessment)
- If a group of systems share threats and risks then their practices and controls should be similar and the assurance evaluation should also be similar (patterns for systems)

Contact Information



Carol Woody, Ph.D.

ccwoody@att.net

Web Resources

www.sei.cmu.edu/go/cybersecurity-engineering

http://www.sei.cmu.edu/