Secure by Design

Carnegie Mellon University Software Engineering Institute

Timothy A. Chick
CERT Applied Systems Group Technical Manager, CMU-Software Engineering Institute
Adjunct Faculty Member, CMU-Software and Societal Systems Department (S3D)

Document Markings

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.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

DM23-0168

Today: Program Office Whac-A-Mole

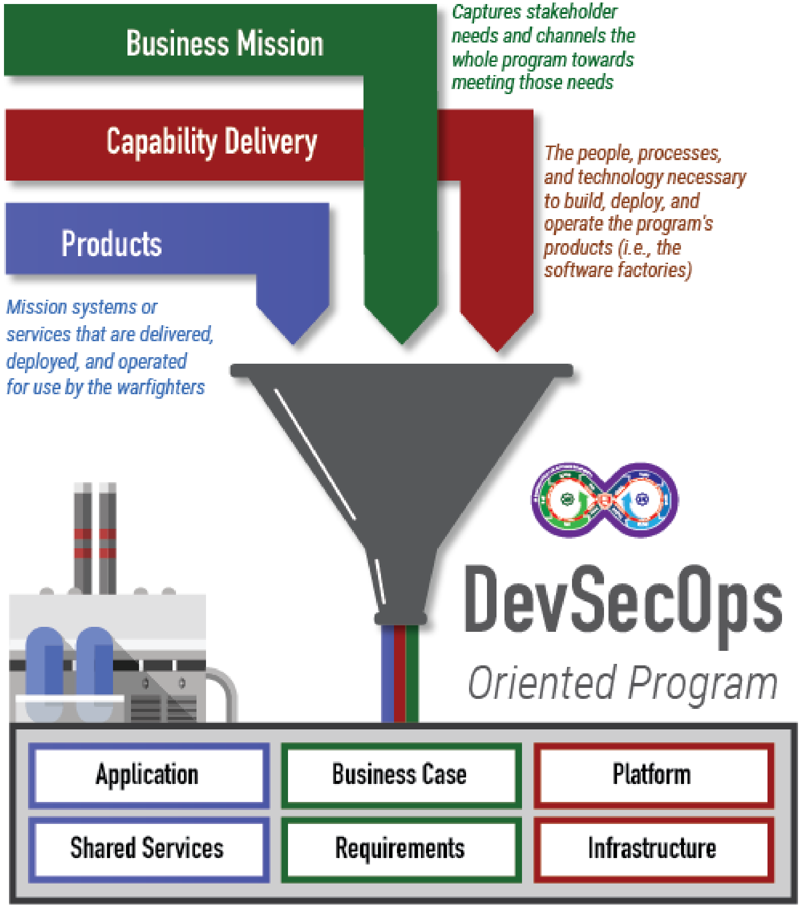
Winning in Features and Effectiveness, but Losing in Defensibility and Stability

In June of 2020 a generally successful DoD program completed an **8 week “Hardening the Software Factory” effort** in order to address **accumulated technical debt** and to address **insufficient security and operations** practices **due to the narrow focus on speed of delivery**.

These things occur, even in small relatively successful programs, when technical debt and insufficient security and operational practices are in place **due to lack of knowledge, experience, and reference material to fully design and execute an integrated DevSecOps strategy in which all stakeholder needs, including cybersecurity, are addressed.**

While playing Whac-A-Mole is inevitable, instead of missing the holes, or constantly hitting the same hole, the key is to fill in the holes.

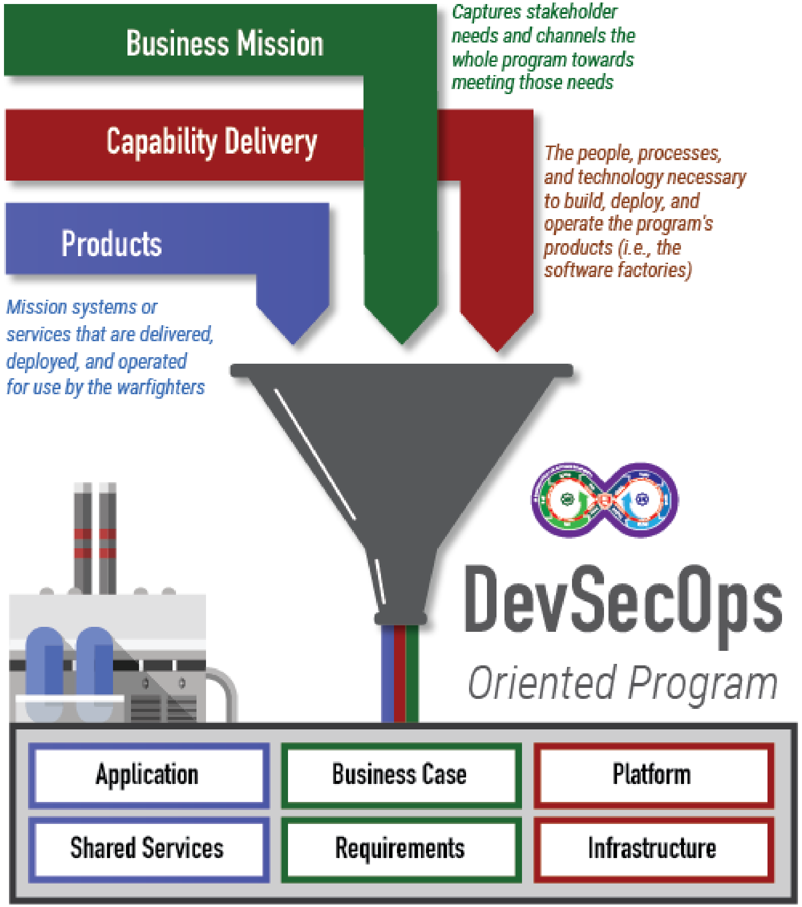
A Program View


Business Mission
Captures stakeholder needs and channels the whole program towards meeting those needs
Capability Delivery
The people, processes, and technology necessary to build, deploy, and operate the program's products (i.e., the software factories)
Products
Mission systems or services that are delivered, deployed, and operated for use by the warfighters
DevSecOps
Oriented Program
Application
Business Case
Platform
Shared Services
Requirements
Infrastructure

All software oriented programs are driven by three concerns:

- **Business Mission** – captures stakeholder needs and channels the whole program in meeting those needs. It answer the questions *Why* and *For Whom* the program exists
- **Capability to Deliver Value** – covers the people, processes, and technology necessary to build, deploy, and operate the program's products
- **Products** – the units of value delivered by the program. Products utilize the capabilities delivered by the software factory and operational environments.

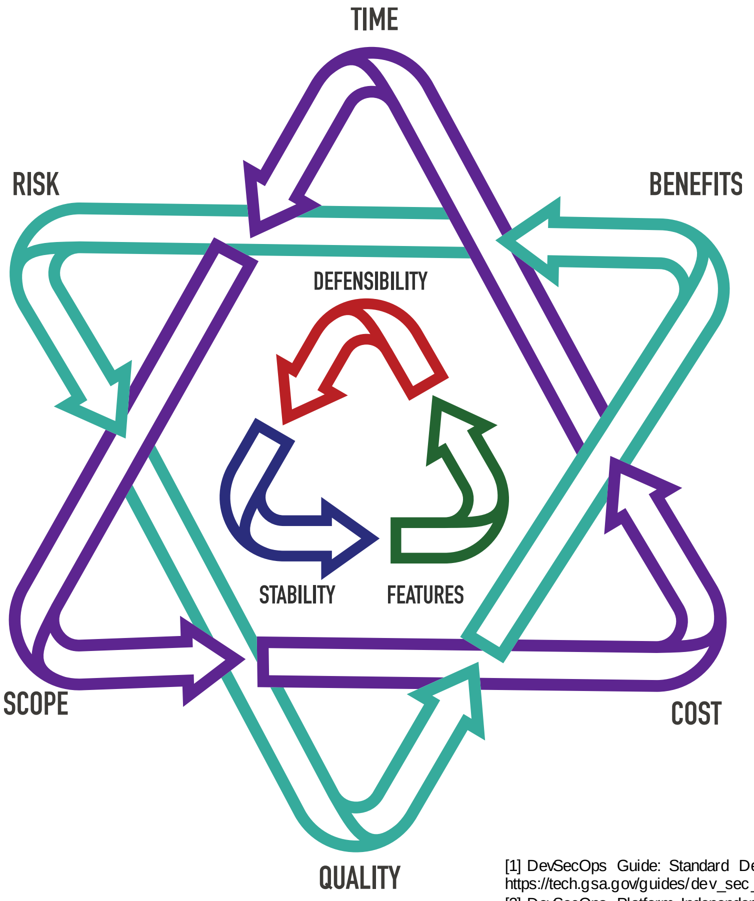
Challenge 1: connecting process, practice, and tools


Business Mission
Captures stakeholder needs and channels the whole program towards meeting those needs
Capability Delivery
The people, processes, and technology necessary to build, deploy, and operate the program's products (i.e., the software factories)
Products
Mission systems or services that are delivered, deployed, and operated for use by the warfighters
DevSecOps
Oriented Program
Application
Business Case
Platform
Shared Services
Requirements
Infrastructure

Capabilities and Products are not static.

- Infrastructure and shared services are often maintained across multiple organizations
- Processes, practices, and tools must evolve to meet the needs of the products being built and operated
- Products must evolve to meet changing needs, defects found, and changes to other systems.

DevSecOps: Modern Software Engineering Practices and Tools that Encompass the Full Software Lifecycle



DevSecOps is a cultural and **engineering practice** that breaks down barriers and opens **collaboration between development, security, and operations** organizations **using automation** to focus on rapid, frequent delivery of secure infrastructure and software to production. It encompasses intake to release of software and manages those flows predictably, transparently, and with minimal human intervention/effort [1].

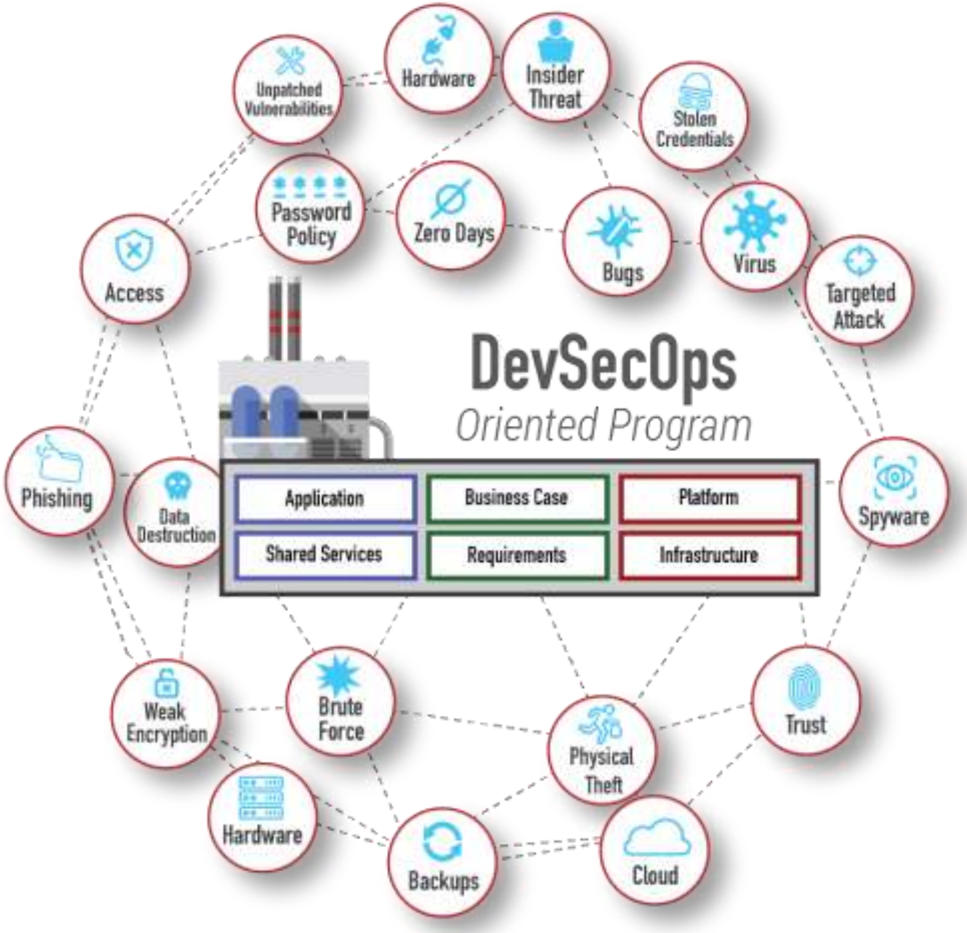
A **DevSecOps Pipeline** attempts to seamlessly integrate "three traditional factions that sometimes have opposing interests:

- **development**; which values features;
- **security**, which values defensibility; and
- **operations**, which values stability [2]."

Not only does one need to balance the factions. They must do so in a way that balances **risk**, **quality** and **benefits** within their **time**, **scope**, and **cost** constraints.

[1] DevSecOps Guide: Standard DevSecOps Platform Framework. U.S. General Services Administration. https://tech.gsa.gov/guides/dev_sec_ops_guide. Accessed 17 May 2021
[2] DevSecOps Platform Independent Model, https://cmu-sei.github.io/DevSecOps-Model/

Challenge 2: Addressing Threats to both Pipeline and Product



The tight integration of Business Mission, Capability Delivery, and Products, using integrated processes, tools, and people, increases the attack surface of the product under development.

Managing and monitoring all the various parts to ensure the product is built with sufficient cybersecurity and the pipeline is maintained to operate with sufficient cybersecurity is complex.

How do you focus attention to areas of greatest concern for security risks and identify the attack opportunities that could require additional mitigations?

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

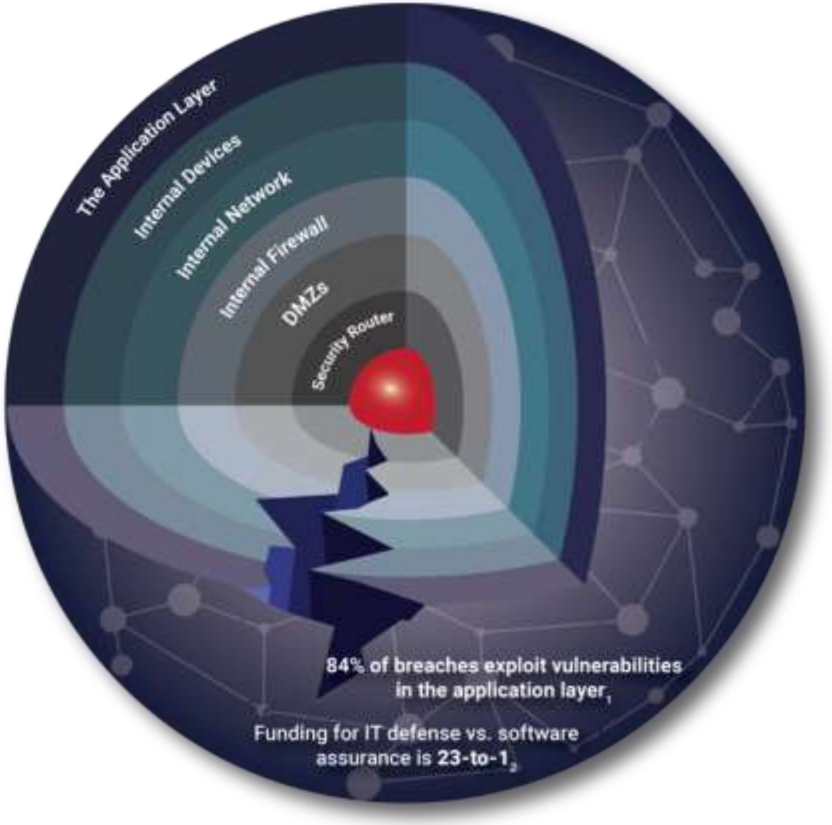
Using a capability service to attack a product isn't new

https://www.itworld.com/article/2861675/cyberattack-on-german-steel-factory-causes-massive-damage.html

"**Steelworks compromise causes massive damage to furnace.**

One of the most concerning was a targeted APT attack on a German steelworks which ended in the attackers gaining access to the business systems and through them to the production network (including SCADA). The effect was that the attackers gained control of a steel furnace and this caused massive damages to the plant."

One Opening is all an Adversary Needs


The Application Layer
Internal Devices
Internal Network
Internal Firewall
DMZs
Security Router
84% of breaches exploit vulnerabilities in the application layer1
Funding for IT defense vs. software assurance is 23-to-12

The Application Layer is the new perimeter exploited by 84% of breaches

Security must be Engineered into the Lifecycle of Applications changing the way we build and buy technology

1. Clark, Tim, *Most cyber Attacks Occur from this Common Vulnerability,* Forbes. 03-10-2015
2. Feiman, Joseph, *Maverick Research: Stop Protecting Your Apps; It's Time for Apps to Protect Themselves,* Gartner. 09-25-2014. G00269825

Software Assurance (SwA)

DoD definition:

“the level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at anytime during its lifecycle, and that the **software functions in the intended manner**.”

[CNSS Instruction No. 4009; DoDi 5200.44 p.12]

SwA Curriculum Model definition:

Application of technologies and processes to achieve a required level of confidence that **software systems and services function in the intended manner**, are free from accidental or intentional vulnerabilities, provide security capabilities appropriate to the threat environment, and recover from intrusions and failures.

[Mead, Nancy; Allen, Julia; Ardis, Mark; Hilburn, Thomas; Kornecki, Andrew; Linger, Richard; & McDonald, James. *Software Assurance Curriculum Project Volume I: Master of Software Assurance Reference Curriculum.* CMU/SEI-2010-TR-005. Software Engineering Institute, Carnegie Mellon University. 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9415]

Risk

The perception of risk drives assurance decisions

- Assurance implementation choices (policies, practices, tools, restrictions) are based on the perception of threat and the expected impact should that threat be realized
- Perceptions are primarily based on knowledge about successful attacks
  - the current state of assurance is largely reactive
  - successful organizations learn from attacks and figure out how to react and recover faster and be vigilant in anticipating and detecting attacks
- Misperceptions are failures to recognize threats and impacts – “how could it happen to us?” or “it could not happen here!”

Mitigating Risk with Assurance Cases

Understanding risk is hard!

Without being able to quantify, or reason around, the cybersecurity risks associated with your product and DevSecOps pipeline, you will not be able to:

- properly balance between features, defensibility, and stability
- make necessary trade-off choices to achieve your organization’s mission and vision in a cost-effective way

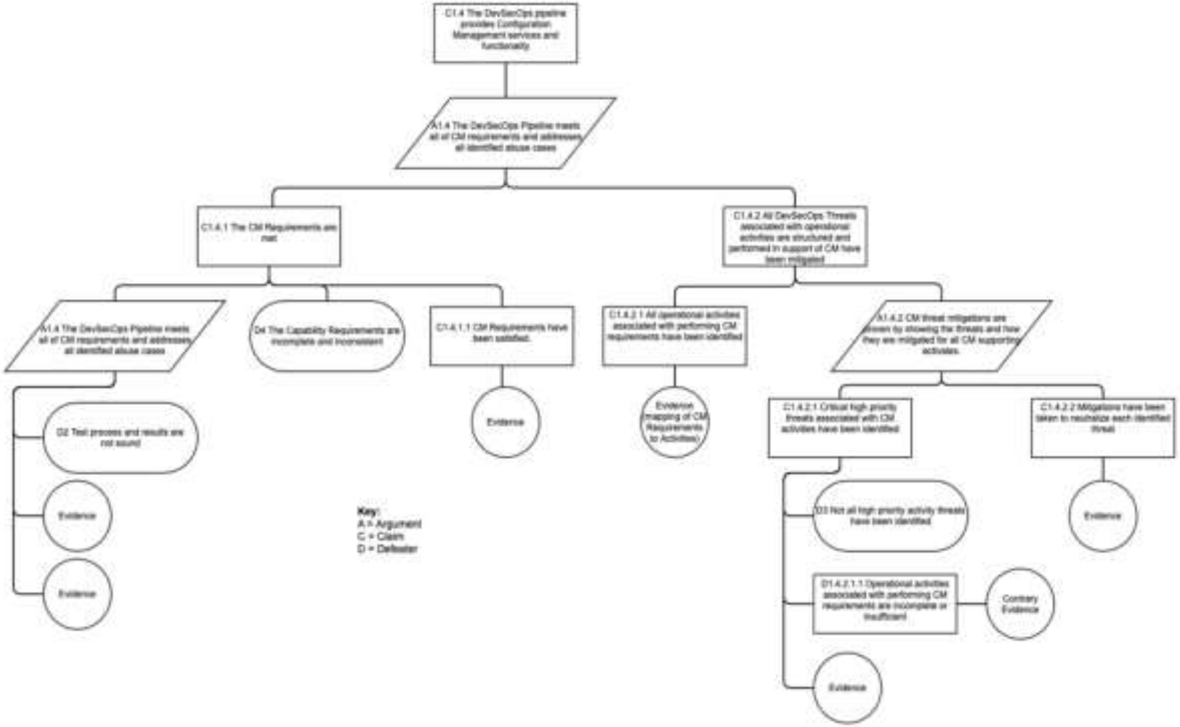
An assurance case can be used to reason about the adequacy for both the pipeline and the product.

- It is a structured approach used to argue that available evidence supports a given claim
- It provides the organization with the basis for making risk-based choices tied to assuring that the pipeline only functions as intended.
- It provides requirements for automated systems testing, or other evidence collection techniques.
- Actual test results provide the evidence needed to support the assurance claims.

Assuring that your Program only Functions as Intended

Assurance cases are composed of the following elements:

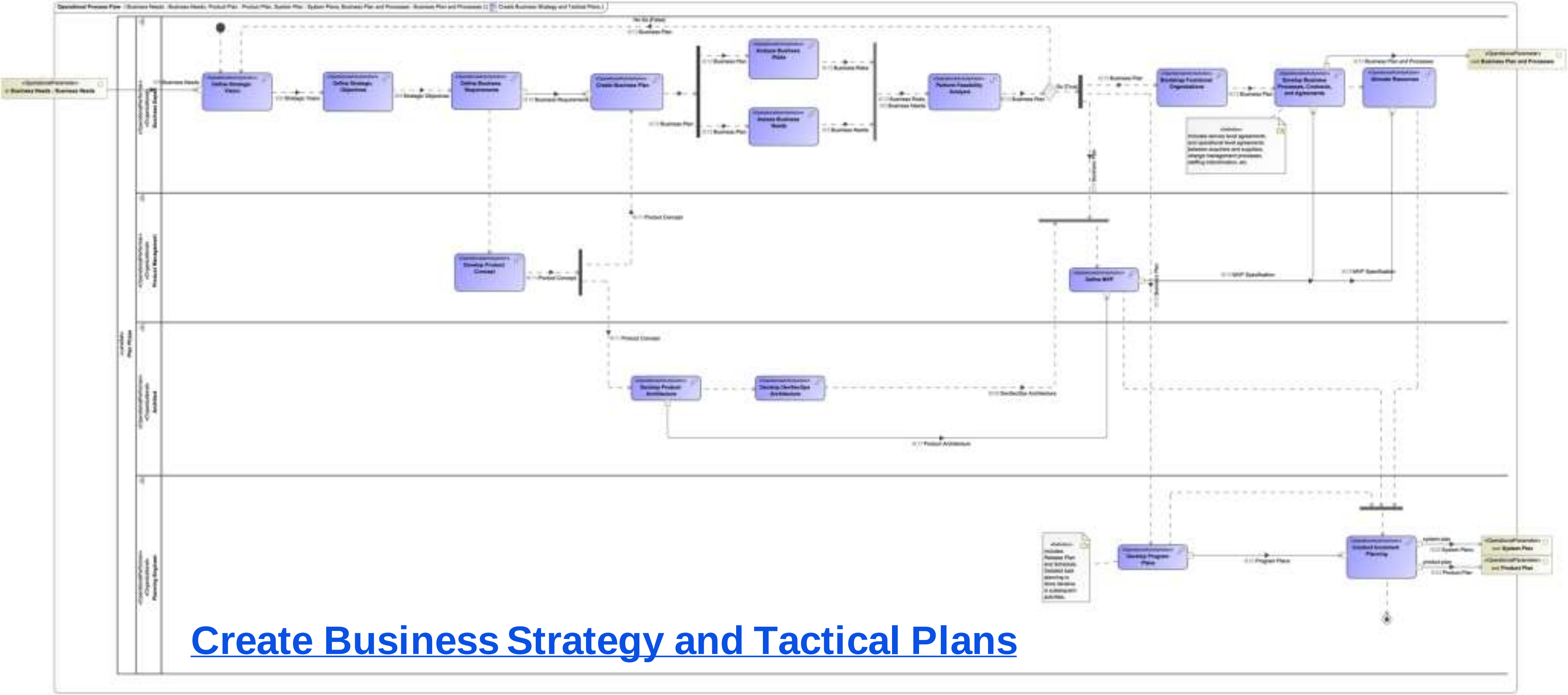
- Claims– "assertions put forward for general acceptance. They are typically statements about a property of the system or some subsystem. Claims that are asserted as true without justification become assumptions and claims supporting an argument are called subclaims [1]."
- Arguments – "link the evidence to the claim [1]" by stating the assumption(s) on which the claim and the evidence are built upon.
- Evidence – "Evidence that is used as the basis of the justification of the claim. Sources of evidence may include the design, the development process, prior field experience, testing, source code analysis or formal analysis [1]."
- Defeaters – "possible reasons for doubting the truth of a claim [2]."



C1.4 The DevSecOps pipeline provides Configuration Management services and functionality
A1.4 The DevSecOps Pipeline meets all of CM requirements and addresses all identified abuse cases
C1.4.1 The CM Requirements are met
C1.4.2 All DevSecOps Threats associated with operational activities are structured and performed in support of CM have been mitigated
A1.4 The DevSecOps Pipeline meets all of CM requirements and addresses all identified abuse cases
D4 The Capability Requirements are incomplete and inconsistent
C1.4.1.1 CM Requirements have been satisfied.
C1.4.2.1 All operational activities associated with performing CM requirements have been identified
A1.4.2 CM threat mitigations are shown by showing the threats and how they are mitigated for all CM supporting activities.
D2 Test process and results are not sound
Evidence
Evidence (mapping of CM Requirements to Activities)
C1.4.2.1 Critical high priority threats associated with CM activities have been identified
C1.4.2.2 Mitigations have been taken to neutralize each identified threat
Evidence
Evidence
Key:
A = Argument
C = Claim
D = Defeater
D3 Not all high priority activity threats have been identified
Evidence
D1.4.2.1.1 Operational activities associated with performing CM requirements are incomplete or insufficient
Contrary Evidence
Evidence

[1] Bloomfield, R. E. and Netkachova, K. Building Blocks for Assurance Cases. Paper presented at the International Symposium on Software Reliability Engineering (ISSRE), 03-11-2014 - 06-11-2014, Naples, Italy.
[2] Goodenough, John B., Charles B. Weinstock, Ari Z. Klein. Toward a Theory of Assurance Case Confidence, CMU/SEI-2012-TR-002 September 2012.

Security Starts at Inception

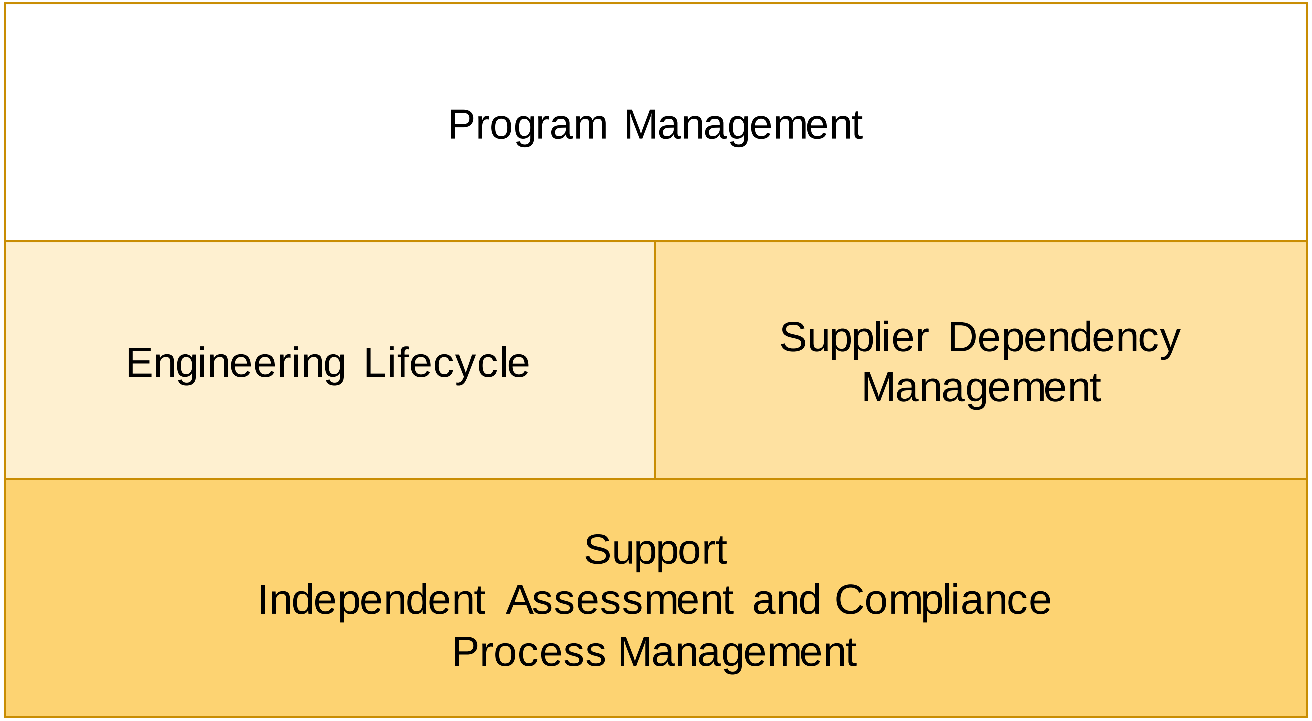


Create Business Strategy and Tactical Plans

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Acquisition Security Framework (ASF)

Acquisition Security Framework (ASF)



Program Management

Engineering Lifecycle

Supplier Dependency Management

Support
Independent Assessment and Compliance
Process Management

Four of the six areas are ready for use: Program Management, Engineering Lifecycle, Supplier Dependency Management, and Support. The remaining areas have been drafted and will be completed this calendar year.

What is the ASF?

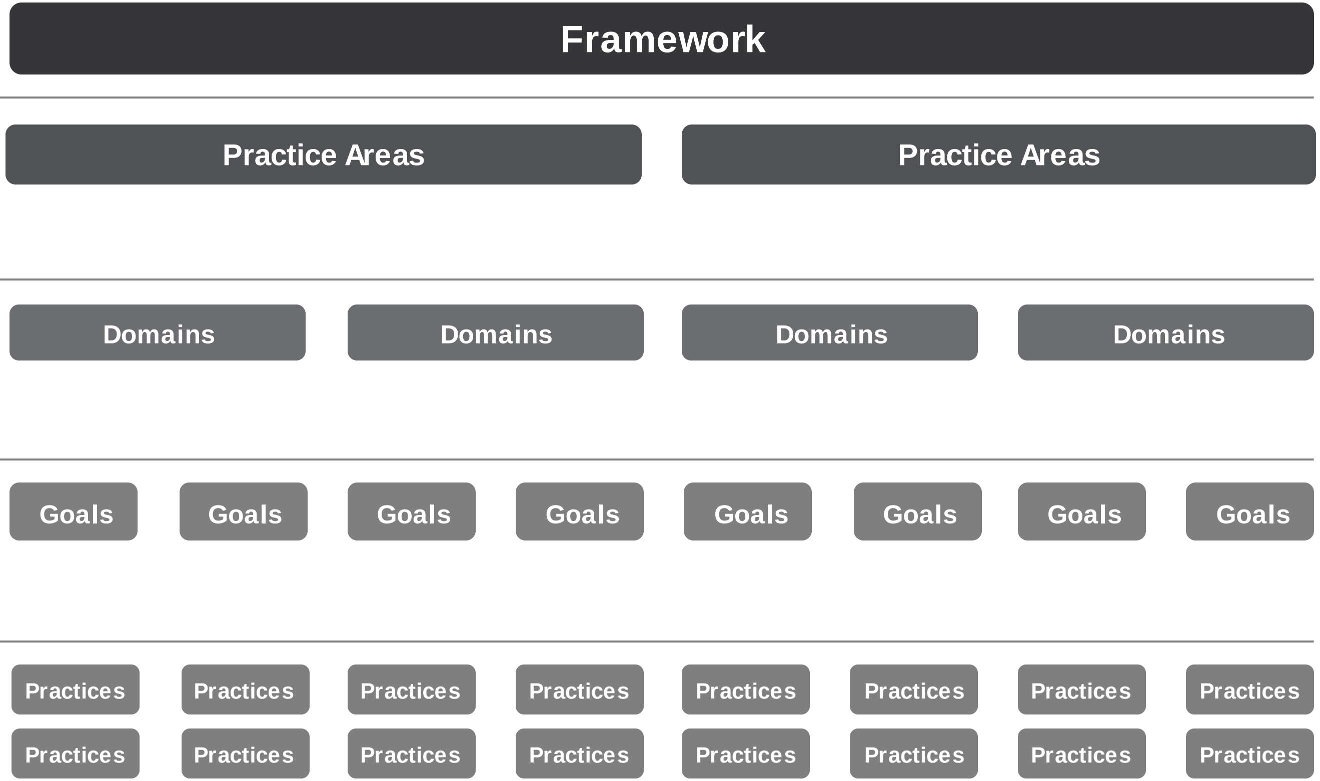
The Acquisition Security Framework (ASF) is a collection of leading practices for building and operating secure and resilient software-reliant systems.

The ASF is designed to proactively enable system security and resilience engineering across the lifecycle and supply chain.

ASF provides a roadmap for building security and resilience into a system rather than attempting to “bolt it on” after deployment.

ASF facilitates efficient and predictable systems environments and more manageable delivery and risk outcomes.

ASF Structure



Framework

The framework comprises multiple practice areas.

Practice Areas

Practice Areas

Each practice area comprises multiple domains

Domains

Domains

Domains

Domains

Each domain comprises multiple goals.

Goals

Goals

Goals

Goals

Goals

Goals

Goals

Goals

Each capability comprises multiple practices

Practices

Practices

Practices

Practices

Practices

Practices

Practices

Practices

Practices

Practices

Practices

Practices

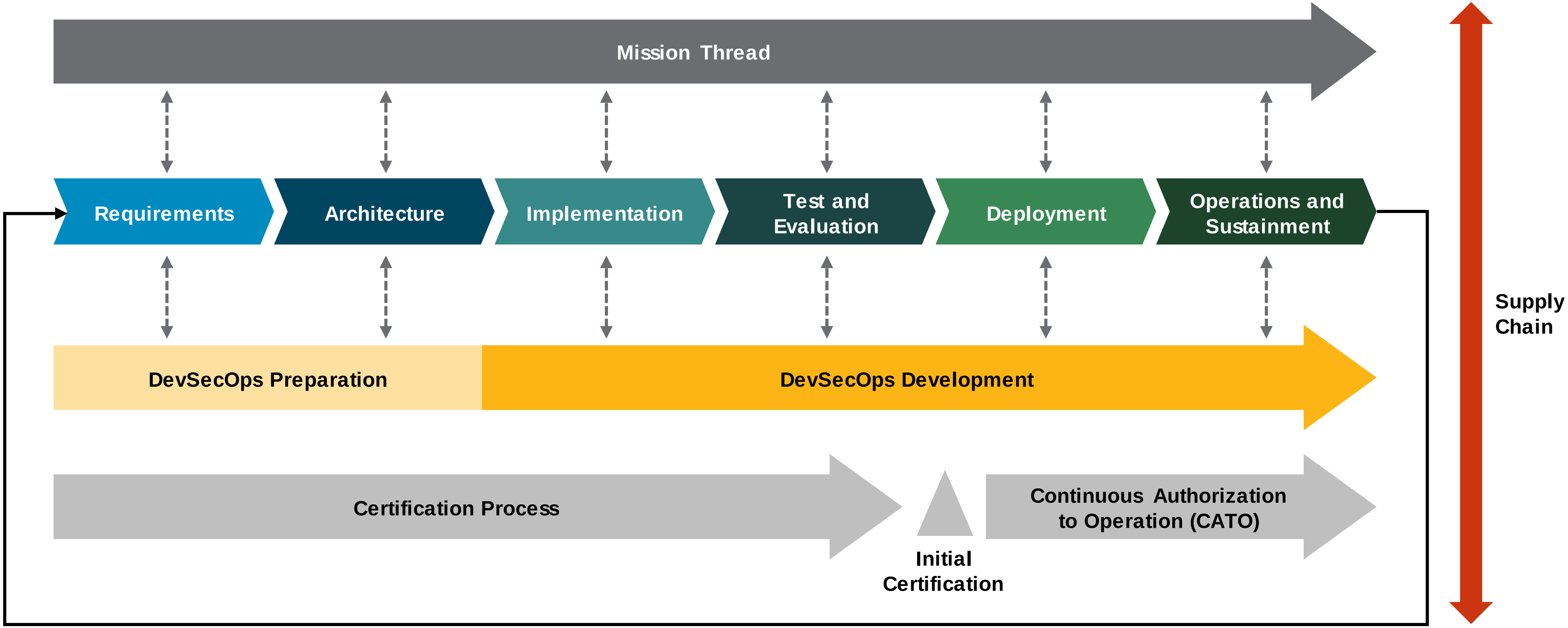
Practices

Practices

Practices

Practices

Cybersecurity Problem Space



Mission Thread

Requirements

Architecture

Implementation

Test and Evaluation

Deployment

Operations and Sustainment

DevSecOps Preparation

DevSecOps Development

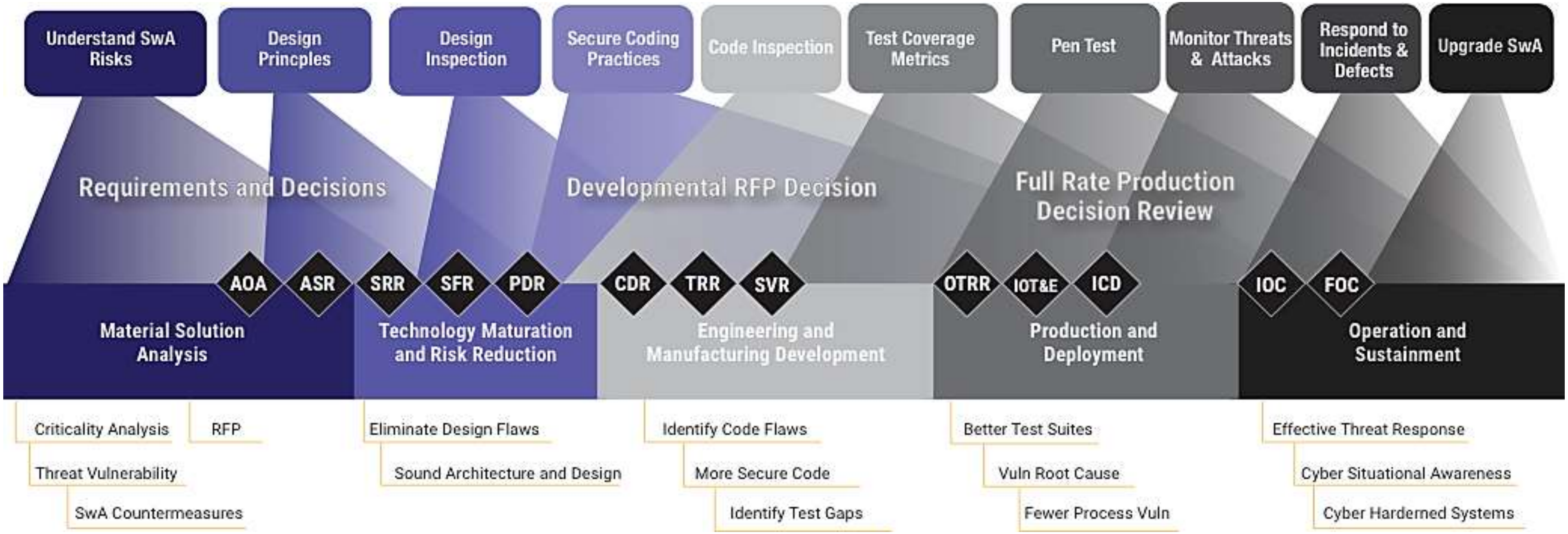
Certification Process

Initial Certification

Continuous Authorization to Operation (CATO)

Supply Chain

Just like Quality, Security is a lifecycle challenge



Security Requirements Challenges

Typical problems with security requirements

- Stated as specific security solutions (practices) and not real requirements
  - Ex: Only authorized users shall access personal healthcare information
- Too narrowly focused on security in a particular application
  - Ex: use SSL for Web communication
- Compliance mandates are substituted for security requirements
  - Ex: An audit log must be maintained of every access to the patient’s healthcare information
- Focused on selection of controls after designs are complete
- Ignored in requirements elicitation because no stakeholders are knowledgeable enough about security impacts to state their security requirements

Merely Specifying Security Features is Insufficient

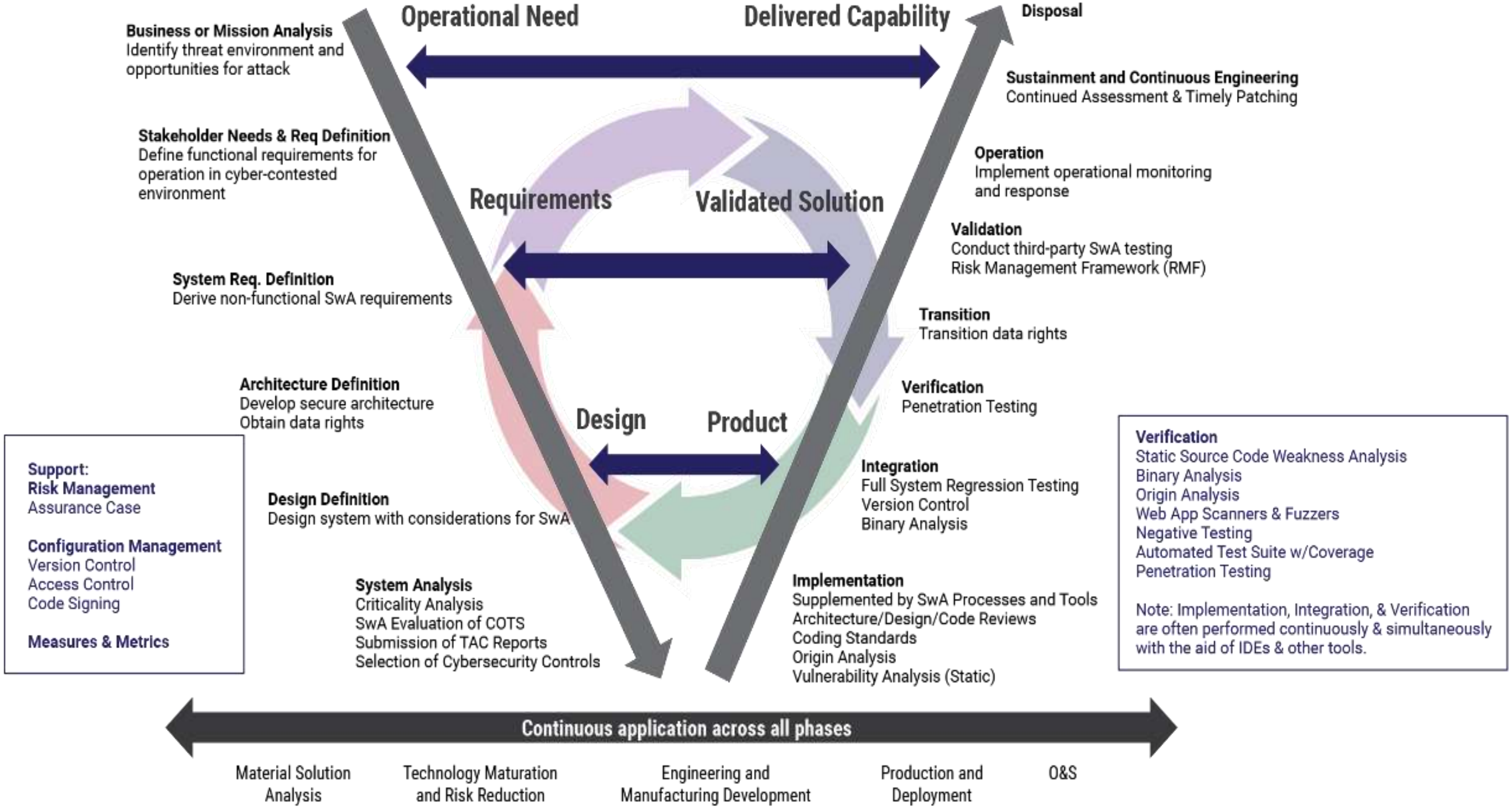
One needs to

- anticipate ways in which a system can be misused by adversaries
- perform systematic, rigorous, and customized threat analysis
- associate attack methods with the likely identified threats
- define and document mitigation strategies aimed at thwarting the attacks
- Write appropriately specific security requirements

“*Early specification of security requirements positively impacts fundamental architectural decisions that enable security concerns to be addressed from the ground up, rather than added as late-in-the-day patches in an attempt to remediate security vulnerabilities.*”

https://resources.sei.cmu.edu/asset_files/TechnicalNote/2018_004_001_516627.pdf

Software Assurance Activities Mapping



Operational Need

Delivered Capability

Requirements

Validated Solution

Design

Product

Business or Mission Analysis
Identify threat environment and opportunities for attack

Stakeholder Needs & Req Definition
Define functional requirements for operation in cyber-contested environment

System Req. Definition
Derive non-functional SwA requirements

Architecture Definition
Develop secure architecture
Obtain data rights

Design Definition
Design system with considerations for SwA

System Analysis
Criticality Analysis
SwA Evaluation of COTS
Submission of TAC Reports
Selection of Cybersecurity Controls

Implementation
Supplemented by SwA Processes and Tools
Architecture/Design/Code Reviews
Coding Standards
Origin Analysis
Vulnerability Analysis (Static)

Integration
Full System Regression Testing
Version Control
Binary Analysis

Verification
Penetration Testing

Transition
Transition data rights

Validation
Conduct third-party SwA testing
Risk Management Framework (RMF)

Operation
Implement operational monitoring and response

Sustainment and Continuous Engineering
Continued Assessment & Timely Patching

Disposal

Support:
Risk Management
Assurance Case

Configuration Management
Version Control
Access Control
Code Signing

Measures & Metrics

Verification
Static Source Code Weakness Analysis
Binary Analysis
Origin Analysis
Web App Scanners & Fuzzers
Negative Testing
Automated Test Suite w/Coverage
Penetration Testing

Note: Implementation, Integration, & Verification are often performed continuously & simultaneously with the aid of IDEs & other tools.

Continuous application across all phases

Material Solution Analysis

Technology Maturation and Risk Reduction

Engineering and Manufacturing Development

Production and Deployment

O&S

Threat Modeling

- **Threat Modeling** is the process of creating an abstraction of a system, aimed at identifying attackers' abilities and goals, and using that abstraction to generate and catalog possible threats that the system must mitigate.
- While security can be analyzed at the networking and code levels to prevent buffer overflows, SQL injection attacks, etc. there is value in **creating a mindset of defensive thinking** early in the requirements and architecture phases.
- **Defensive thinking** means that for every new feature, one must think about how it could be abused or defeated by adversaries.
- The defensive thinking mindset **underlies the approach to threat modeling**

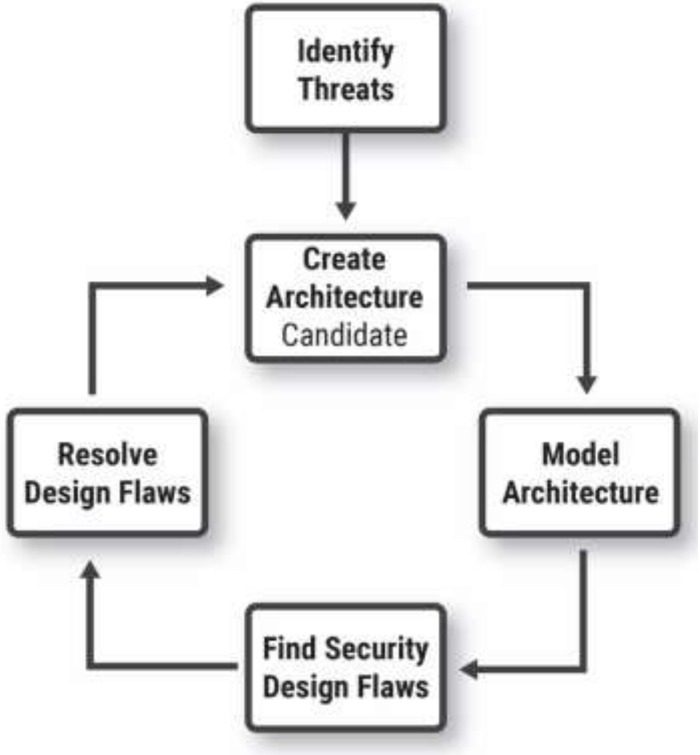
https://insights.sei.cmu.edu/blog/the-hybrid-threat-modeling-method/

Value of Modeling Security

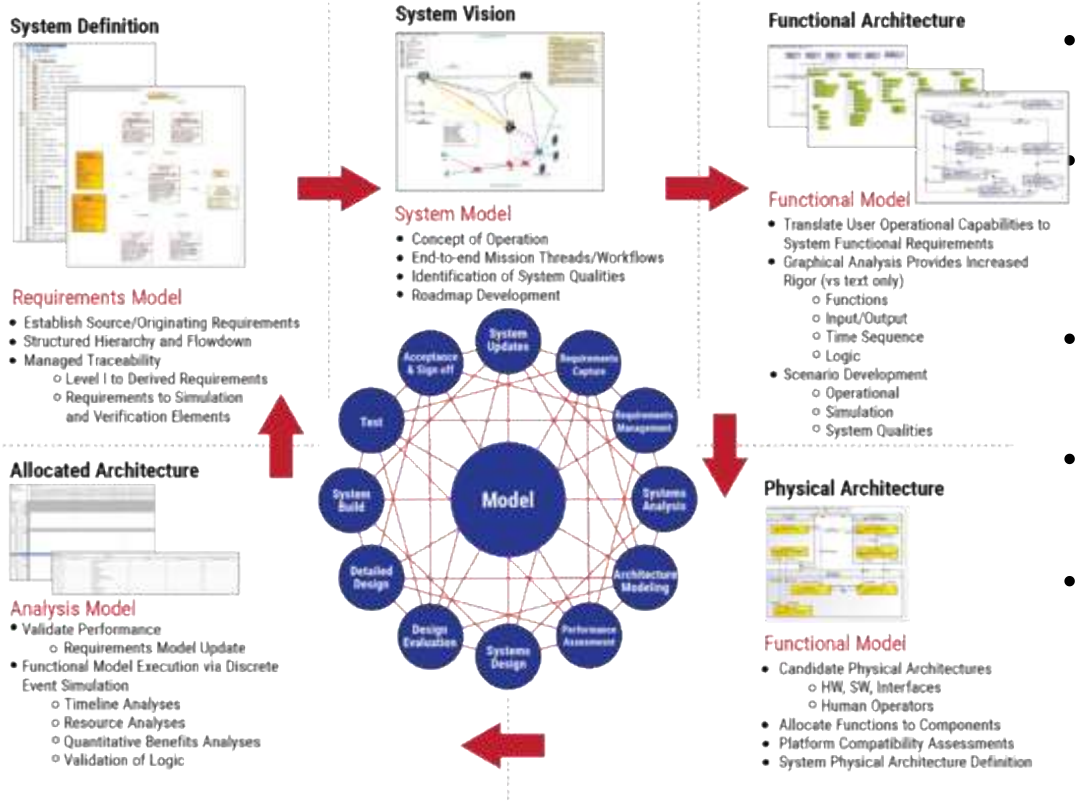
Crucial security decisions to address threats are made in the architecture.

Analyzing an architecture is a huge opportunity for improving security.

Threat Modeling methods can be combined with MBSE to create a more robust and well-rounded view of potential threats.



Model Based Systems Engineering



Requirements Model

- Establish Source/Originating Requirements
- Structured Hierarchy and Flowdown
- Managed Traceability
  - Level I to Derived Requirements
  - Requirements to Simulation and Verification Elements

System Model

- Concept of Operation
- End-to-end Mission Threads/Workflows
- Identification of System Qualities
- Roadmap Development

Functional Model

- Translate User Operational Capabilities to System Functional Requirements
- Graphical Analysis Provides Increased Rigor (vs text only)
  - Functions
  - Input/Output
  - Time Sequence
  - Logic
- Scenario Development
  - Operational
  - Simulation
  - System Qualities

Functional Model

- Candidate Physical Architectures
  - HW, SW, Interfaces
  - Human Operators
- Allocate Functions to Components
- Platform Compatibility Assessments
- System Physical Architecture Definition

Analysis Model

- Validate Performance
  - Requirements Model Update
- Functional Model Execution via Discrete Event Simulation
  - Timeline Analyses
  - Resource Analyses
  - Quantitative Benefits Analyses
  - Validation of Logic

*The Digital System Model contains the most current requirements, key mission/business operations, architecture, design details, implementation details, test and evaluation details, and supporting documentation.

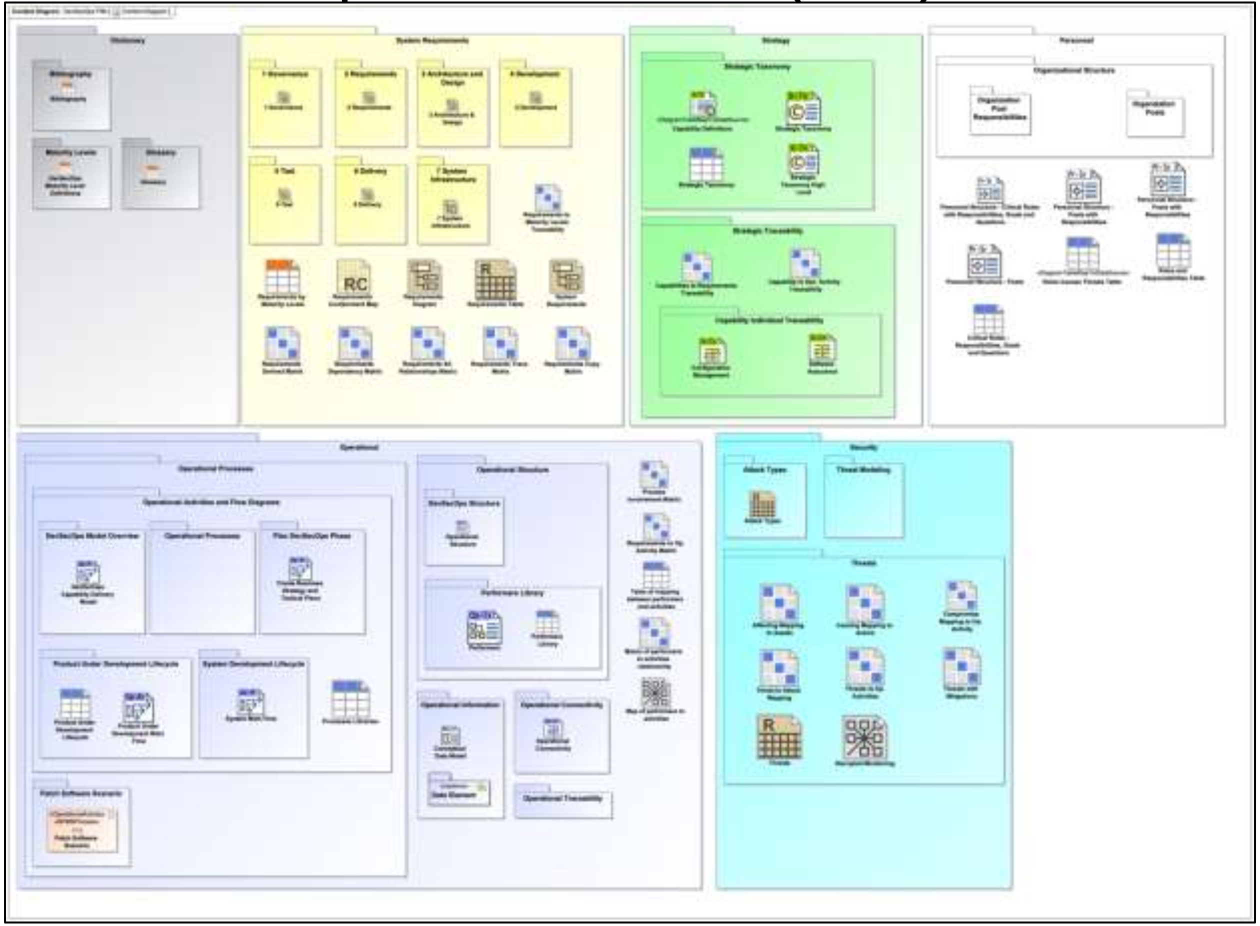
- ***Not* yesterday's Document-Centric Systems Engineering!**
- MBSE uses a Digital System Model* to facilitate common system understanding and decision-making.
- The Digital System Model* is the single authoritative source of truth
- System and Components can be integrated at various levels of abstraction and fidelity
- Model Views are chosen to best communicate information to a variety of stakeholders via the dynamic creation of multiple, consistent, accurate views
- Impacts of changes are more easily analyzed and evaluated

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

A DevSecOps Example

Carnegie Mellon University Software Engineering Institute

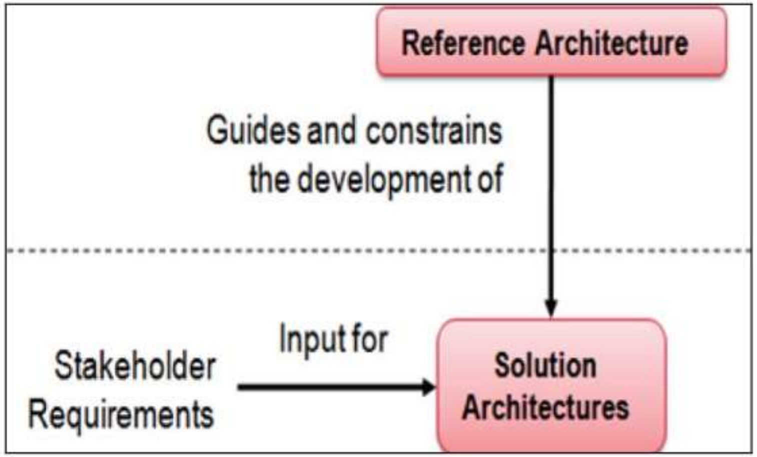
DevSecOps Platform Independent Model (PIM) - Content Diagram



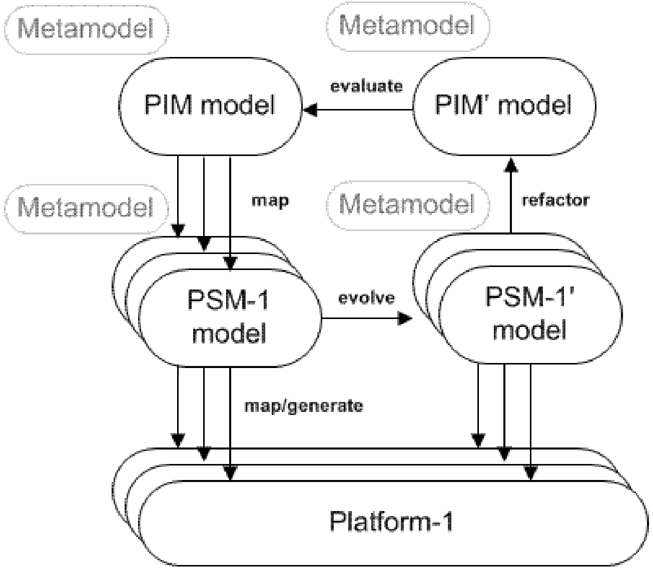
https://cmu-sei.github.io/DevSecOps-Model/

Reference Architecture/Platform Independent Model (PIM)

A **Reference Architecture** is an authoritative source of information about a specific subject area that guides and constrains the instantiations of multiple architectures and solutions [1].



A PIM is a general and reusable model of a solution to a commonly occurring problem in software engineering within a given context and is independent of the specific technological platform used to implement it.



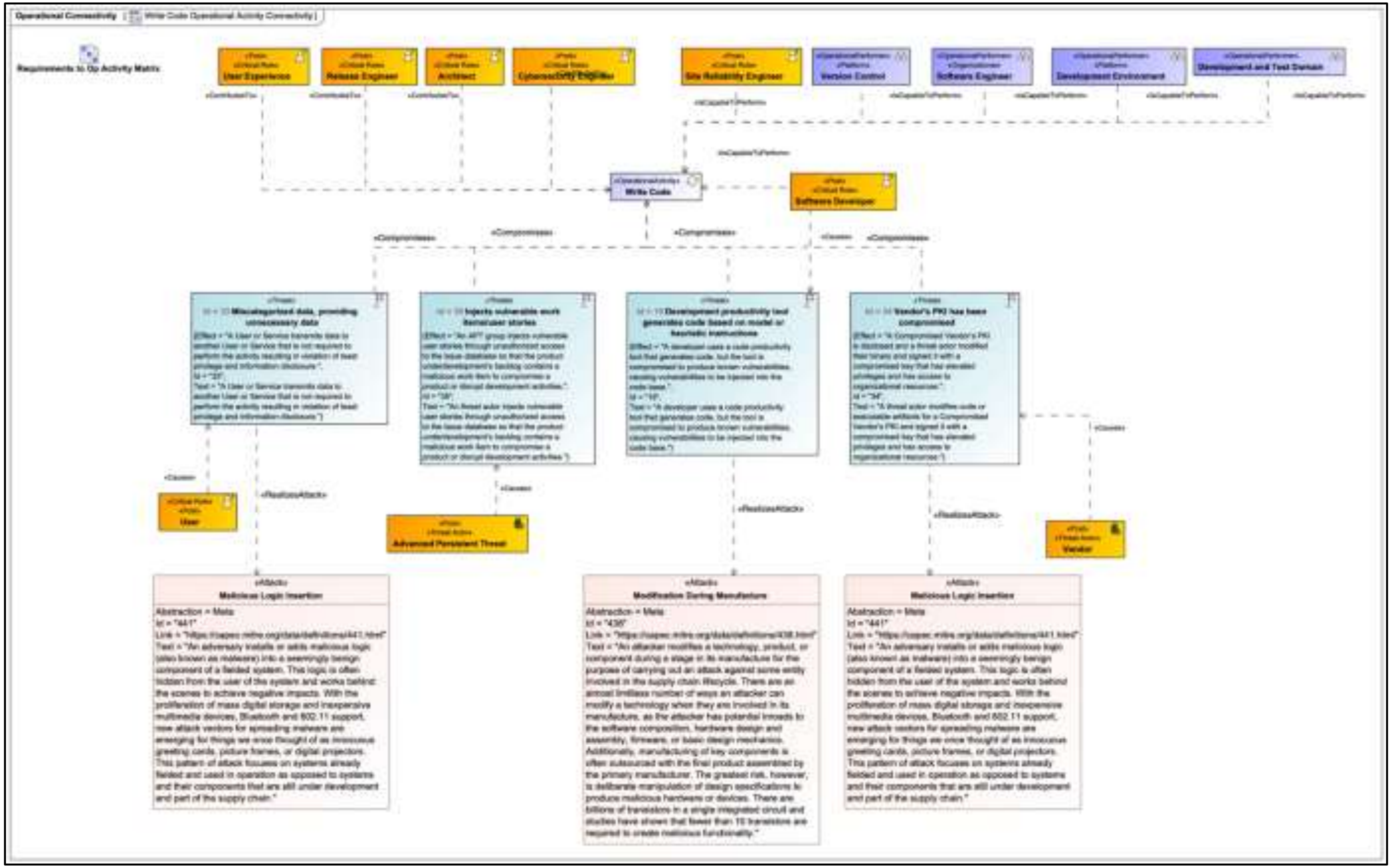
NOTE: PSM = Platform Specific Model

[1] DoD Reference Architecture Description, https://dodcio.defense.gov/Portals/0/Documents/DIEA/Ref_Archi_Description_Final_v1_18Jun10.pdf

The DevSecOps PIM enables Organizations, Projects, Teams, and Acquirers to

- specify the DevSecOps requirements to the lead system integrators tasked with developing a platform-specific solution that includes the designed system and continuous integration/continuous deployment (CI/CD) pipeline
- assess and analyze alternative pipeline functionality and feature changes as the system evolves
- apply DevSecOps methods to complex products that do not follow well-established software architectural patterns used in industry
- provide a basis for threat and attack surface analysis to build a cyber assurance case to demonstrate that the product and DevSecOps pipeline are sufficiently free from vulnerabilities and that they function only as intended

Example Threat Modeling Diagram for Write Code Operational Activity


Operational Connectivity [Write Code Operational Activity Connectivity]
Requirements to Op Activity Matrix
«Role» «Critical Role» User Experience
«Role» «Critical Role» Release Engineer
«Role» «Critical Role» Architect
«Role» «Critical Role» Cybersecurity Engineer
«Role» «Critical Role» Site Reliability Engineer
«OperationalPerformer» «Platform» Version Control
«OperationalPerformer» «Organization» Software Engineer
«OperationalPerformer» «Platform» Development Environment
«OperationalPerformer» Development and Test Domain
«ContributesTo»
«ContributesTo»
«ContributesTo»
«IsCapableToPerform»
«IsCapableToPerform»
«IsCapableToPerform»
«IsCapableToPerform»
«IsCapableToPerform»
«IsCapableToPerform»
«OperationalActivity» Write Code
«Role» «Critical Role» Software Developer
«Compromises»
«Compromises»
«Compromises»
«Causes»
«Compromises»
«Threat»
«Causes»
«RealizesAttack»
«Causes»
«RealizesAttack»
«RealizesAttack»
«Causes»
«Critical Role» «Role» User
«Role» «Threat Actor» Advanced Persistent Threat
«Role» «Threat Actor» Vendor
«Attack»

Id = 23 Miscategorized data, providing unnecessary data

{Effect = "A User or Service transmits data to another User or Service that is not required to perform the activity resulting in violation of least privilege and information disclosure",
Id = "23",
Text = "A User or Service transmits data to another User or Service that is not required to perform the activity resulting in violation of least privilege and information disclosure"}

Id = 38 Injects vulnerable work items/user stories

{Effect = "An APT group injects vulnerable user stories through unauthorized access to the issue database so that the product underdevelopment's backlog contains a malicious work item to compromise a product or disrupt development activities.",
Id = "38",
Text = "An threat actor injects vulnerable user stories through unauthorized access to the issue database so that the product underdevelopment's backlog contains a malicious work item to compromise a product or disrupt development activities."}

Id = 19 Development productivity tool generates code based on model or heuristic instructions

{Effect = "A developer uses a code productivity tool that generates code, but the tool is compromised to produce known vulnerabilities, causing vulnerabilities to be injected into the code base.",
Id = "19",
Text = "A developer uses a code productivity tool that generates code, but the tool is compromised to produce known vulnerabilities, causing vulnerabilities to be injected into the code base."}

Id = 34 Vendor's PKI has been compromised

{Effect = "A Compromised Vendor's PKI is disclosed and a threat actor modified their binary and signed it with a compromised key that has elevated privileges and has access to organizational resources.",
Id = "34",
Text = "A threat actor modifies code or executable artifacts for a Compromised Vendor's PKI and signed it with a compromised key that has elevated privileges and has access to organizational resources."}

Malicious Login Insertion

Abstraction = Meta
Id = "441"
Link = "https://capec.mitre.org/data/definitions/441.html"
Text = "An adversary installs or adds malicious logic (also known as malware) into a seemingly benign component of a fielded system. This logic is often hidden from the user of the system and works behind the scenes to achieve negative impacts. With the proliferation of mass digital storage and inexpensive multimedia devices, Bluetooth and 802.11 support, new attack vectors for spreading malware are emerging for things we once thought of as innocuous greeting cards, picture frames, or digital projectors. This pattern of attack focuses on systems already fielded and used in operation as opposed to systems and their components that are still under development and part of the supply chain."

Modification During Manufacture

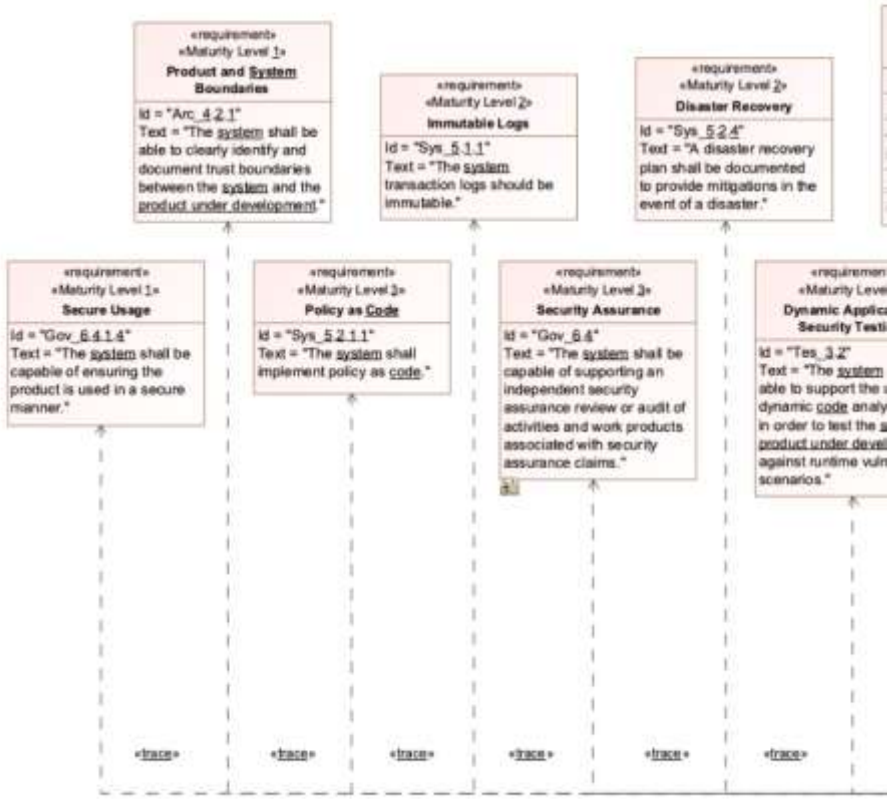
Abstraction = Meta
Id = "438"
Link = "https://capec.mitre.org/data/definitions/438.html"
Text = "An attacker modifies a technology, product, or component during a stage in its manufacture for the purpose of carrying out an attack against some entity involved in the supply chain lifecycle. There are an almost limitless number of ways an attacker can modify a technology when they are involved in its manufacture, as the attacker has potential inroads to the software composition, hardware design and assembly, firmware, or basic design mechanics. Additionally, manufacturing of key components is often outsourced with the final product assembled by the primary manufacturer. The greatest risk, however, is deliberate manipulation of design specifications to produce malicious hardware or devices. There are billions of transistors in a single integrated circuit and studies have shown that fewer than 10 transistors are required to create malicious functionality."

Malicious Logic Insertion

Abstraction = Meta
Id = "441"
Link = "https://capec.mitre.org/data/definitions/441.html"
Text = "An adversary installs or adds malicious logic (also known as malware) into a seemingly benign component of a fielded system. This logic is often hidden from the user of the system and works behind the scenes to achieve negative impacts. With the proliferation of mass digital storage and inexpensive multimedia devices, Bluetooth and 802.11 support, new attack vectors for spreading malware are emerging for things we once thought of as innocuous greeting cards, picture frames, or digital projectors. This pattern of attack focuses on systems already fielded and used in operation as opposed to systems and their components that are still under development and part of the supply chain."

Write Code Operational Activity Connectivity Link

Requirements



«requirement»
«Maturity Level 1»
Product and System Boundaries
Id = "Arc_4.2.1"
Text = "The system shall be able to clearly identify and document trust boundaries between the system and the product under development."

«requirement»
«Maturity Level 2»
Immutable Logs
Id = "Sys_5.1.1"
Text = "The system transaction logs should be immutable."

«requirement»
«Maturity Level 2»
Disaster Recovery
Id = "Sys_5.2.4"
Text = "A disaster recovery plan shall be documented to provide mitigations in the event of a disaster."

«requirement»
«Maturity Level 1»
Secure Usage
Id = "Gov_6.4.1.4"
Text = "The system shall be capable of ensuring the product is used in a secure manner."

«requirement»
«Maturity Level 3»
Policy as Code
Id = "Sys_5.2.1.1"
Text = "The system shall implement policy as code."

«requirement»
«Maturity Level 3»
Security Assurance
Id = "Gov_6.4"
Text = "The system shall be capable of supporting an independent security assurance review or audit of activities and work products associated with security assurance claims."

«requiremen
«Maturity Leve
**Dynamic Applic
Security Testi**
Id = "Tes_3.2"
Text = "The system
able to support the
dynamic code analy
in order to test the
product under devel
against runtime vuln
scenarios."

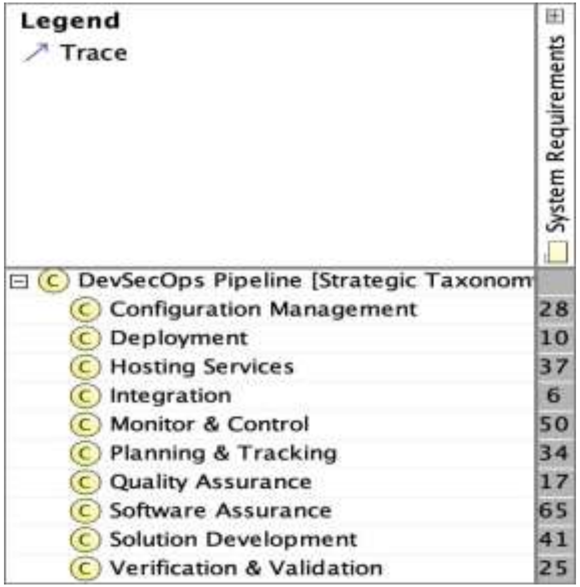
Example of Requirements Representation in Diagrams from PIM

Requirements are organized into categories based on logical and functional groupings

Requirements Table Link

Capability/Strategic Viewpoint

A capability is a high-level concept that describes the ability of a system to achieve or perform a task or a mission.

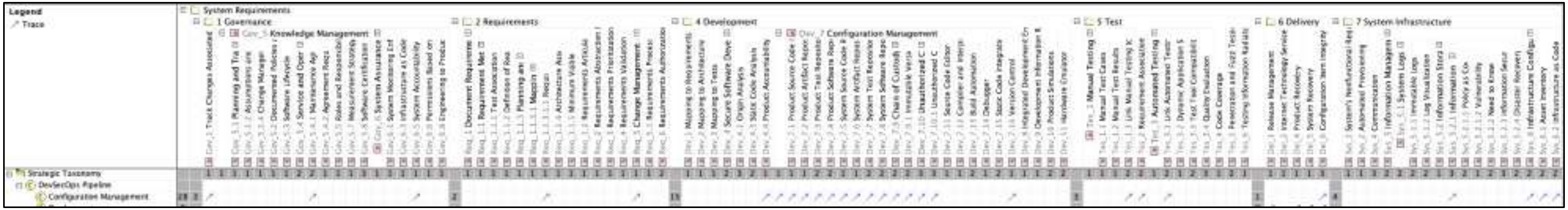


Legend

↗ Trace

DevSecOps Pipeline [Strategic Taxonom	System Requirements
Configuration Management	28
Deployment	10
Hosting Services	37
Integration	6
Monitor & Control	50
Planning & Tracking	34
Quality Assurance	17
Software Assurance	65
Solution Development	41
Verification & Validation	25

- [Capability to Requirements Traceability Link]
- [Capability to Operational Activity Traceability Link]
- [Capability Definitions Link]
- [Strategic Taxonomy High Level]



Legend

↗ Trace

System Requirements

1 Governance | Gov_5 Knowledge Management | 2 Requirements | 4 Development | Dev_7 Configuration Management | 5 Test | 6 Delivery | 7 System Infrastructure

Strategic Taxonomy

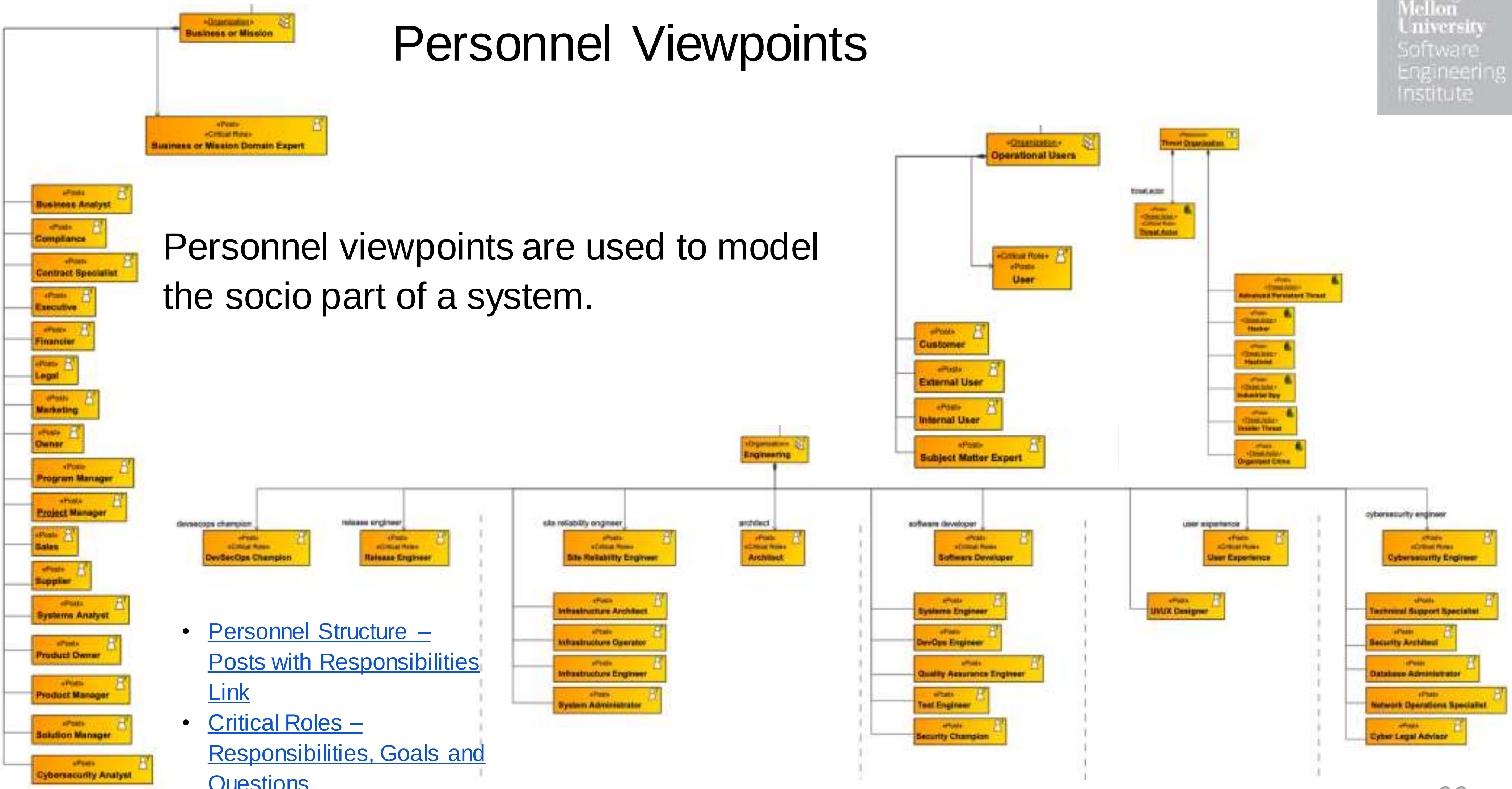
DevSecOps Pipeline

Configuration Management

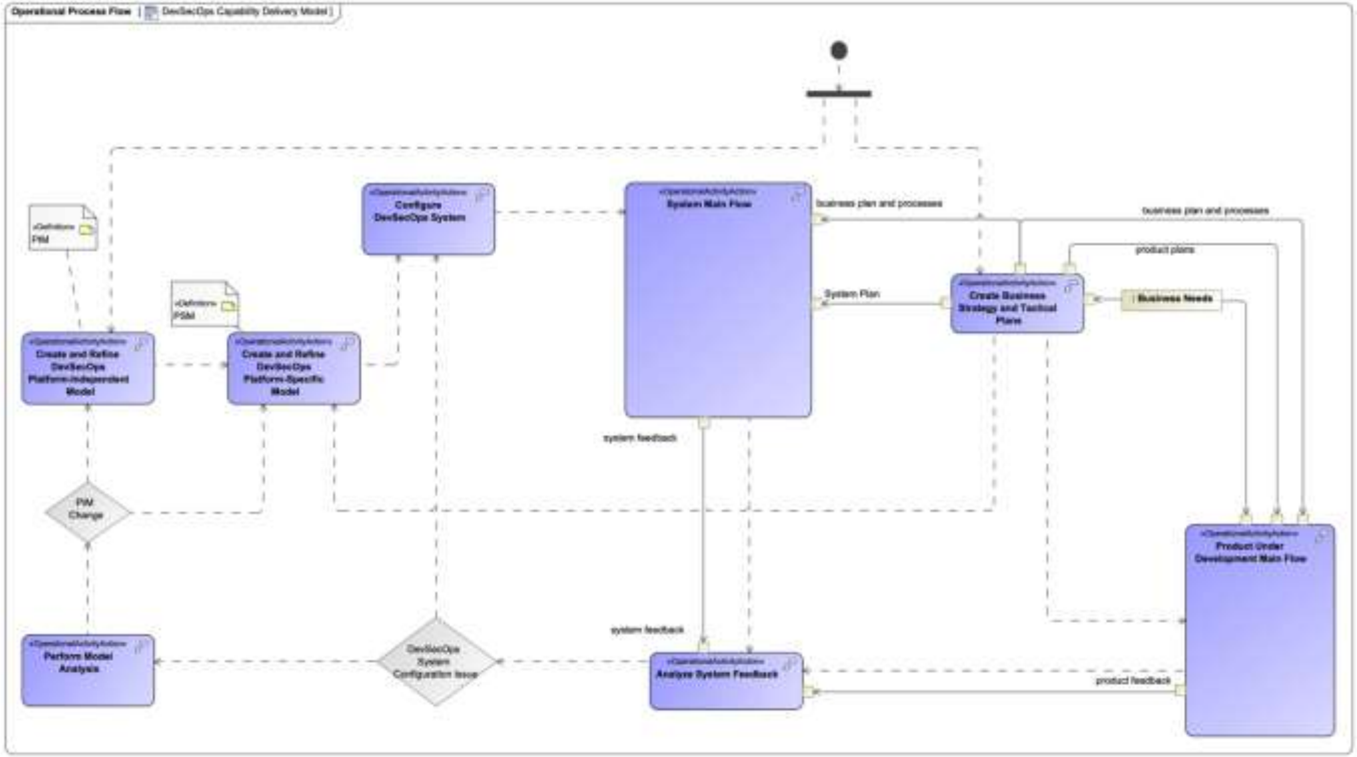
Personnel Viewpoints

Personnel viewpoints are used to model the socio part of a system.

- [Personnel Structure – Posts with Responsibilities Link]
- [Critical Roles – Responsibilities, Goals and Questions]

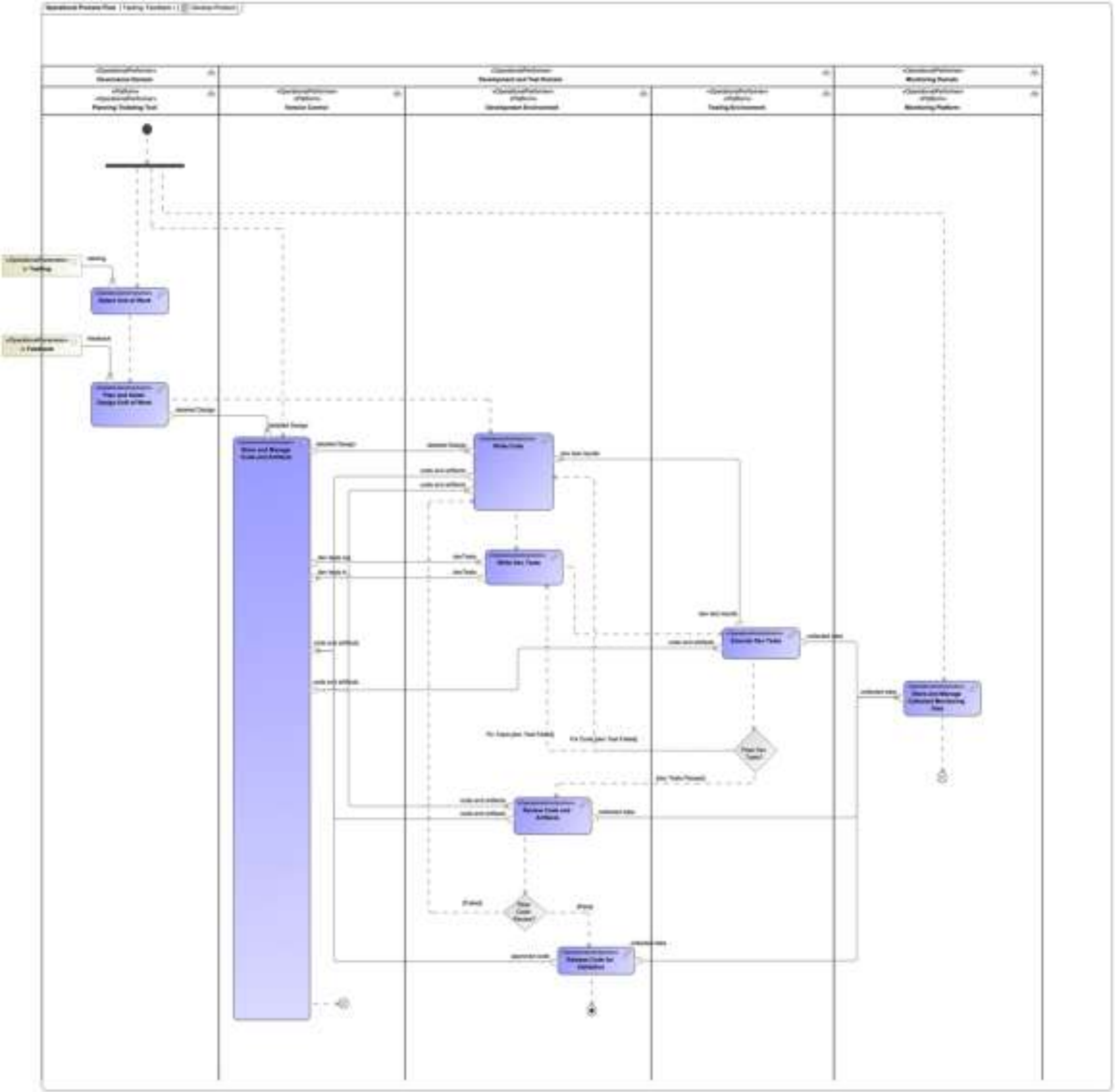


Operational Viewpoints


Operational Process Flow [DevSecOps Capability Delivery Model]
«Definition» PIM
«Definition» PSM
«OperationalActivityAction» Configure DevSecOps System
«OperationalActivityAction» System Main Flow
business plan and processes
business plan and processes
product plans
System Plan
«OperationalActivityAction» Create Business Strategy and Tactical Plans
Business Needs
«OperationalActivityAction» Create and Refine DevSecOps Platform-Independent Model
«OperationalActivityAction» Create and Refine DevSecOps Platform-Specific Model
system feedback
PIM Change
«OperationalActivityAction» Product Under Development Main Flow
system feedback
«OperationalActivityAction» Perform Model Analysis
DevSecOps System Configuration Issue
«OperationalActivityAction» Analyze System Feedback
product feedback

- [DevSecOps Capability Delivery Model Link](#)

An operational model for a system describes behavior of the system to conduct program operations



Operational Process Flow Focus Area

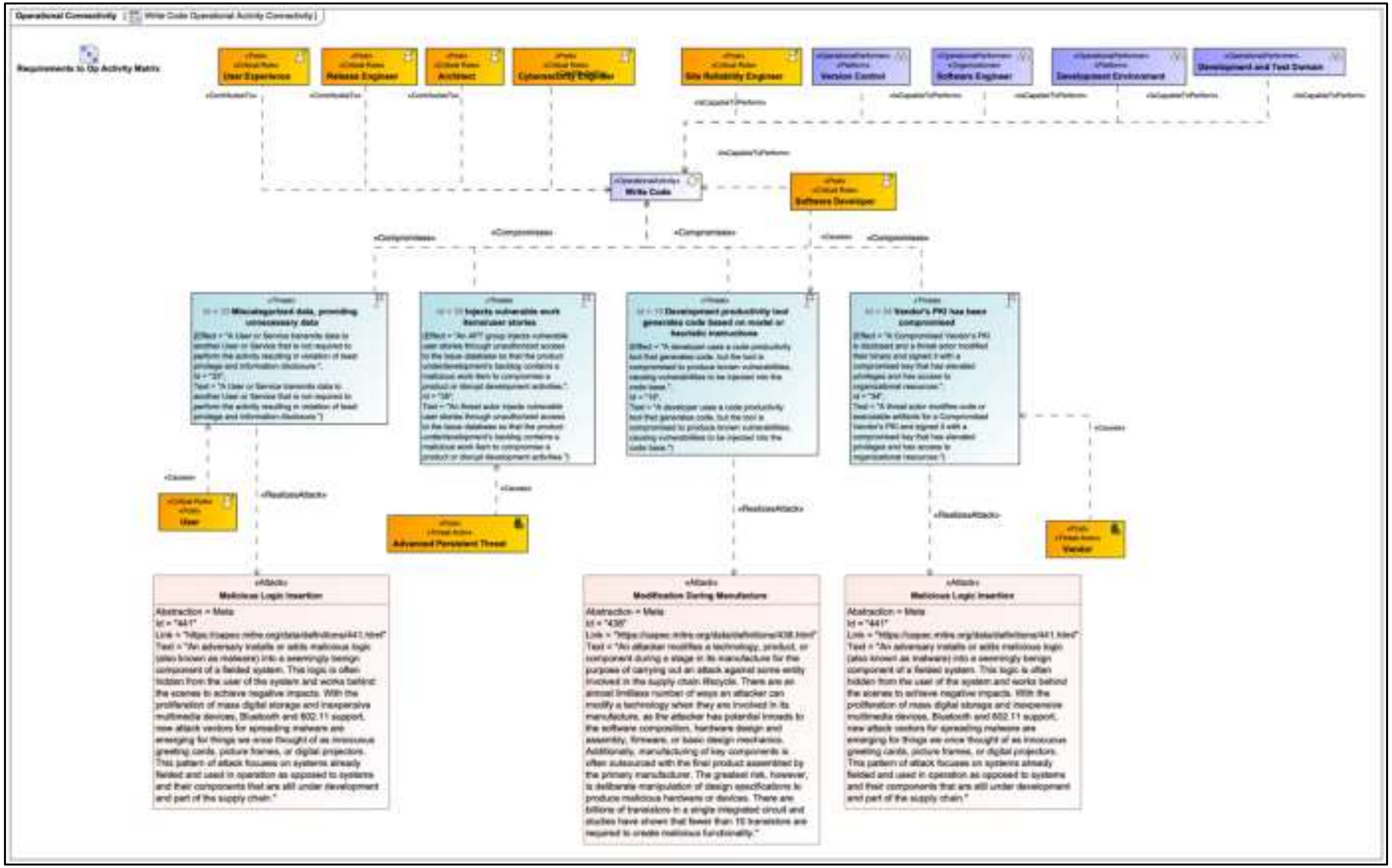
- Select an operational process flow to focus the threat scenario generation
- Review the selected operational process flow to gain understanding of the process, data flow between operational activities, and performers involved
- This may include reviewing associated requirements to understand the scope and context of the various operational activities

Six part Threat Scenario

STATEMENT TEMPLATE: An [ACTOR] performs an [ACTION] to [ATTACK] an [ASSET] to achieve an [EFFECT] and/or [OBJECTIVE].

Part	Description
Actor	The person, or group, that is behind the threat scenario. Threat actors can be malicious or unintentional. Developing a standard set of actors is beneficial for this step. Persona non grata could be useful in determining malicious actors. Threat actor may be a person, or group, internal to an organization structure.
Action	A potential occurrence of an event that might damage an asset, a mission, or goal of a strategic vision.
Attack	An action taken that utilizes one of more vulnerabilities to realize a threat to compromise or damage an asset, a mission, or goal of a strategic vision.
Asset	A resource, person, or process that has value.
Effect	The desired or undesired consequence resulting from the attack.
Objective	The threat actor’s motivation or objective for conducting the attack

Example Threat Modeling Diagram for Write Code Operational Activity


Operational Connectivity [Write Code Operational Activity Connectivity]
Requirements to Op Activity Matrix
«Role» «Critical Role» User Experience
«Role» «Critical Role» Release Engineer
«Role» «Critical Role» Architect
«Role» «Critical Role» Cybersecurity Engineer
«Role» «Critical Role» Site Reliability Engineer
«OperationalPerformer» «Platform» Version Control
«OperationalPerformer» «Organization» Software Engineer
«OperationalPerformer» «Platform» Development Environment
«OperationalPerformer» Development and Test Domain
«ContributesTo»
«ContributesTo»
«ContributesTo»
«IsCapableToPerform»
«IsCapableToPerform»
«IsCapableToPerform»
«IsCapableToPerform»
«IsCapableToPerform»
«IsCapableToPerform»
«OperationalActivity» Write Code
«Role» «Critical Role» Software Developer
«Compromises»
«Compromises»
«Compromises»
«Causes»
«Compromises»
«Threat»
«Threat»
«Threat»
«Threat»
«Causes»
«Critical Role» «Role» User
«RealizesAttack»
«Causes»
«Role» «Threat Actor» Advanced Persistent Threat
«RealizesAttack»
«RealizesAttack»
«Causes»
«Role» «Threat Actor» Vendor
«Attack»
«Attack»
«Attack»

Id = 33 Miscategorized data, providing unnecessary data

(Effect = "A User or Service transmits data to another User or Service that is not required to perform the activity resulting in violation of least privilege and information disclosure",
Id = "33",
Text = "A User or Service transmits data to another User or Service that is not required to perform the activity resulting in violation of least privilege and information disclosure.")

Id = 38 Injects vulnerable work items/user stories

(Effect = "An APT group injects vulnerable user stories through unauthorized access to the issue database so that the product underdevelopment's backlog contains a malicious work item to compromise a product or disrupt development activities.",
Id = "38",
Text = "An threat actor injects vulnerable user stories through unauthorized access to the issue database so that the product underdevelopment's backlog contains a malicious work item to compromise a product or disrupt development activities.")

Id = 11 Development productivity tool generates code based on model or heuristic instructions

(Effect = "A developer uses a code productivity tool that generates code, but the tool is compromised to produce known vulnerabilities, causing vulnerabilities to be injected into the code base.",
Id = "11",
Text = "A developer uses a code productivity tool that generates code, but the tool is compromised to produce known vulnerabilities, causing vulnerabilities to be injected into the code base.")

Id = 34 Vendor's PKI has been compromised

(Effect = "A Compromised Vendor's PKI is disclosed and a threat actor modified their binary and signed it with a compromised key that has elevated privileges and has access to organizational resources.",
Id = "34",
Text = "A threat actor modifies code or executable artifacts for a Compromised Vendor's PKI and signed it with a compromised key that has elevated privileges and has access to organizational resources.")

Malicious Logic Insertion

Abstraction = Meta
Id = "441"
Link = "https://capec.mitre.org/data/definitions/441.html"
Text = "An adversary installs or adds malicious logic (also known as malware) into a seemingly benign component of a fielded system. This logic is often hidden from the user of the system and works behind the scenes to achieve negative impacts. With the proliferation of mass digital storage and inexpensive multimedia devices, Bluetooth and 802.11 support, new attack vectors for spreading malware are emerging for things we once thought of as innocuous greeting cards, picture frames, or digital projectors. This pattern of attack focuses on systems already fielded and used in operation as opposed to systems and their components that are still under development and part of the supply chain."

Modification During Manufacture

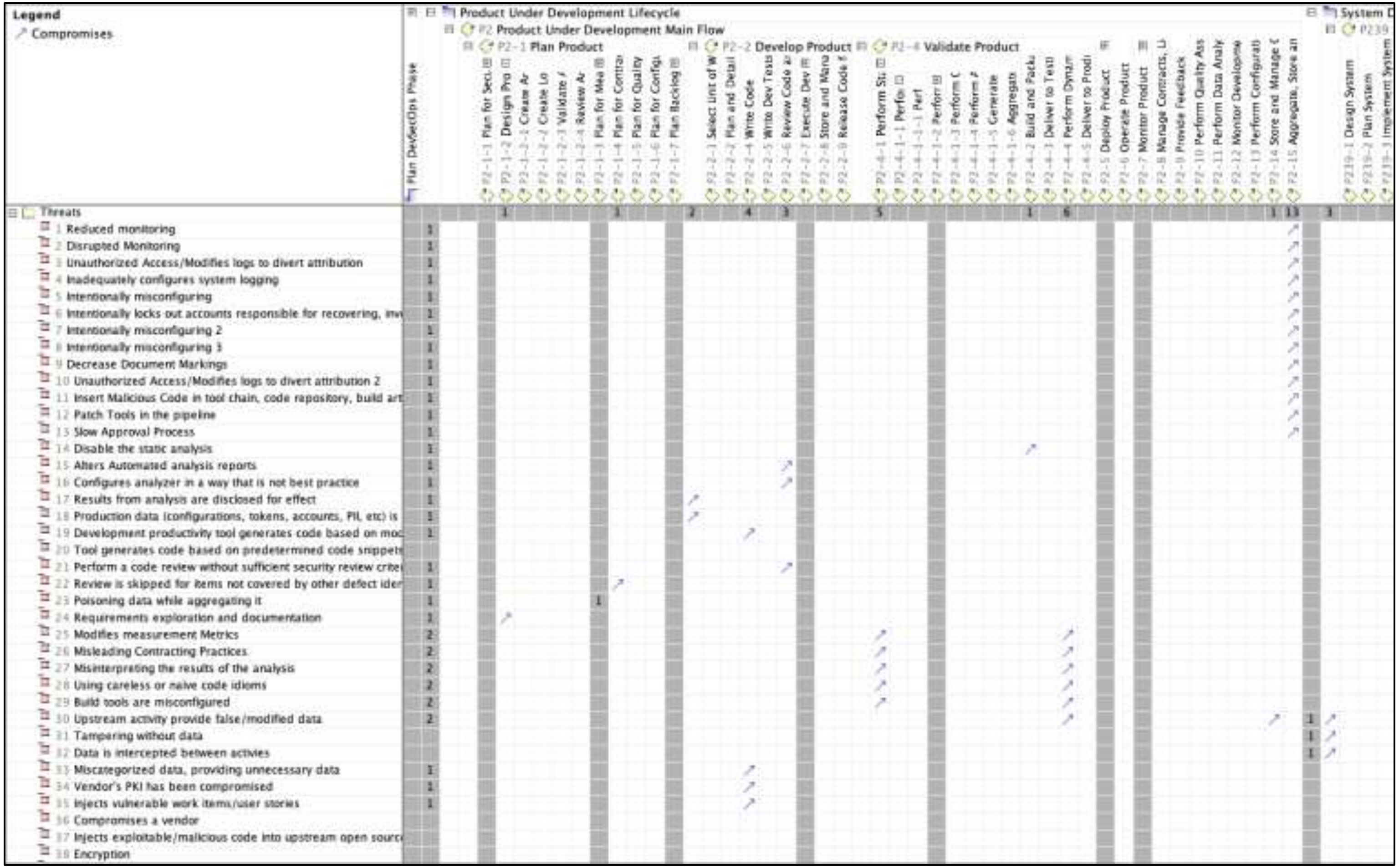
Abstraction = Meta
Id = "438"
Link = "https://capec.mitre.org/data/definitions/438.html"
Text = "An attacker modifies a technology, product, or component during a stage in its manufacture for the purpose of carrying out an attack against some entity involved in the supply chain lifecycle. There are an almost limitless number of ways an attacker can modify a technology when they are involved in its manufacture, as the attacker has potential inroads to the software composition, hardware design and assembly, firmware, or basic design mechanics. Additionally, manufacturing of key components is often outsourced with the final product assembled by the primary manufacturer. The greatest risk, however, is deliberate manipulation of design specifications to produce malicious hardware or devices. There are billions of transistors in a single integrated circuit and studies have shown that fewer than 10 transistors are required to create malicious functionality."

Malicious Logic Insertion

Abstraction = Meta
Id = "441"
Link = "https://capec.mitre.org/data/definitions/441.html"
Text = "An adversary installs or adds malicious logic (also known as malware) into a seemingly benign component of a fielded system. This logic is often hidden from the user of the system and works behind the scenes to achieve negative impacts. With the proliferation of mass digital storage and inexpensive multimedia devices, Bluetooth and 802.11 support, new attack vectors for spreading malware are emerging for things we once thought of as innocuous greeting cards, picture frames, or digital projectors. This pattern of attack focuses on systems already fielded and used in operation as opposed to systems and their components that are still under development and part of the supply chain."

Write Code Operational Activity Connectivity Link

Threat to Operational Activity Matrix



Legend

↗ Compromises

Product Under Development Lifecycle

P2 Product Under Development Main Flow

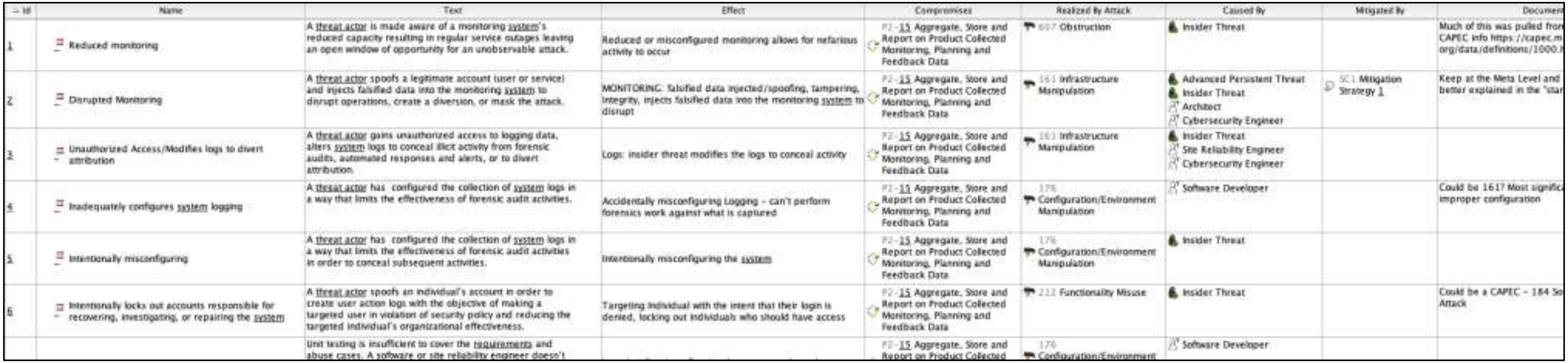
P2-1 Plan Product · P2-2 Develop Product · P2-4 Validate Product

System D · P239

	Plan DevSecOps Phase	P2-1-1 Plan for Secu	P2-1-2 Design Pro	P2-1-2-1 Create Ar	P2-1-2-2 Create Lo	P2-1-2-3 Validate	P2-1-2-4 Review Ar	P2-1-3 Plan for Mea	P2-1-4 Plan for Contra	P2-1-5 Plan for Quality	P2-1-6 Plan for Configu	P2-1-7 Plan Backlog		P2-2-1 Select Unit of W	P2-2-2 Plan and Detail	P2-2-4 Write Code	P2-2-5 Write Dev Tests	P2-2-6 Review Code a	P2-2-7 Execute Dev	P2-2-8 Store and Mana	P2-2-9 Release Code	P2-4-1 Perform St	P2-4-1-1 Perfo	P2-4-1-1-1 Perf	P2-4-1-2 Perform	P2-4-1-3 Perform C	P2-4-1-4 Perform	P2-4-1-5 Generate	P2-4-1-6 Aggregate	P2-4-2 Build and Packa	P2-4-3 Deliver to Test	P2-4-4 Perform Dynam	P2-4-5 Deliver to Prod	P2-5 Deploy Product	P2-6 Operate Product	P2-7 Monitor Product	P2-8 Manage Contracts, Li	P2-9 Provide Feedback	P2-10 Perform Quality Ass	P2-11 Perform Data Analy	P2-12 Monitor Developme	P2-13 Perform Configurati	P2-14 Store and Manage	P2-15 Aggregate, Store an			P239-1 Design System	P239-2 Plan System	P239-3 Implement System
Threats				1						1				2			4		3				5								1		6											1	13		3			
1 Reduced monitoring	1																																												↗					
2 Disrupted Monitoring	1																																												↗					
3 Unauthorized Access/Modifies logs to divert attribution	1																																												↗					
4 Inadequately configures system logging	1																																												↗					
5 Intentionally misconfiguring	1																																												↗					
6 Intentionally locks out accounts responsible for recovering, inv	1																																												↗					
7 Intentionally misconfiguring 2	1																																												↗					
8 Intentionally misconfiguring 3	1																																												↗					
9 Decrease Document Markings	1																																												↗					
10 Unauthorized Access/Modifies logs to divert attribution 2	1																																												↗					
11 Insert Malicious Code in tool chain, code repository, build art	1																																												↗					
12 Patch Tools in the pipeline	1																																												↗					
13 Slow Approval Process	1																																												↗					
14 Disable the static analysis	1																													↗																			
15 Alters Automated analysis reports	1																	↗																															
16 Configures analyzer in a way that is not best practice	1																	↗																															
17 Results from analysis are disclosed for effect	1												↗																																				
18 Production data (configurations, tokens, accounts, PII, etc) is	1												↗																																				
19 Development productivity tool generates code based on moc	1															↗																																	
20 Tool generates code based on predetermined code snippet																																																		
21 Perform a code review without sufficient security review crite	1																	↗																															
22 Review is skipped for items not covered by other defect iden	1								↗																																								
23 Poisoning data while aggregating it	1							1																																									
24 Requirements exploration and documentation	1		↗																																														
25 Modifies measurement Metrics	2																					↗										↗																	
26 Misleading Contracting Practices	2																					↗										↗																	
27 Misinterpreting the results of the analysis	2																					↗										↗																	
28 Using careless or naive code idioms	2																					↗										↗																	
29 Build tools are misconfigured	2																					↗										↗																	
30 Upstream activity provide false/modified data	2																					↗										↗											↗		1	↗			
31 Tampering without data																																													1	↗			
32 Data is intercepted between activies																																													1	↗			
33 Miscategorized data, providing unnecessary data	1															↗																																	
34 Vendor's PKI has been compromised	1															↗																																	
35 Injects vulnerable work items/user stories	1															↗																																	
36 Compromises a vendor																																																		
37 Injects exploitable/malicious code into upstream open sourc																																																		
38 Encryption																																																		

Threats to Operational Activities Link

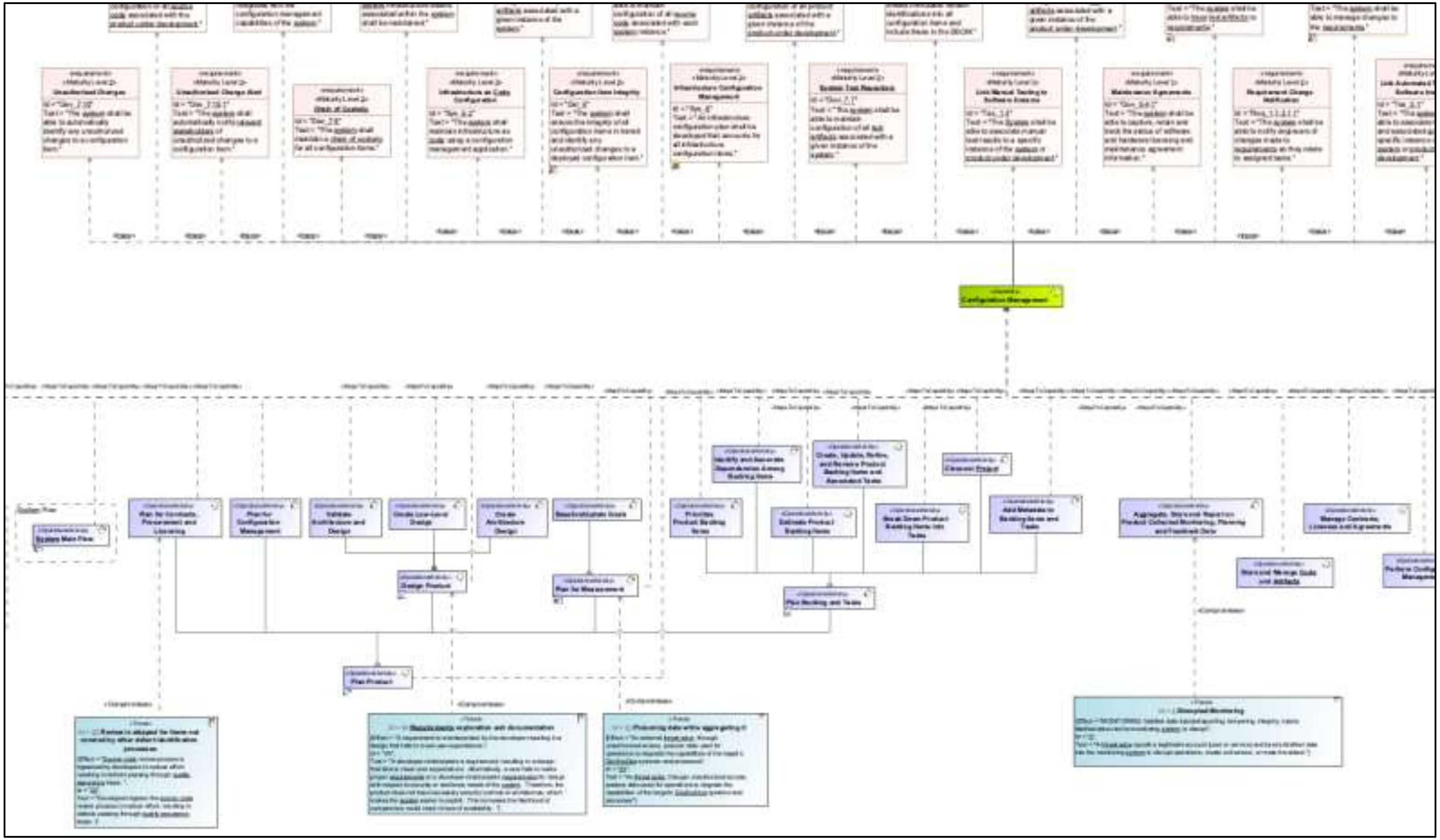
Threats with Attributes



Id	Name	Text	Effect	Compromises	Realized By Attack	Caused By	Mitigated By	Documen
1	Reduced monitoring	A threat actor is made aware of a monitoring system's reduced capacity resulting in regular service outages leaving an open window of opportunity for an unobservable attack.	Reduced or misconfigured monitoring allows for nefarious activity to occur	P2-15 Aggregate, Store and Report on Product Collected Monitoring, Planning and Feedback Data	607 Obstruction	Insider Threat		Much of this was pulled fron CAPEC info https://capec.m org/data/definitions/1000.h
2	Disrupted Monitoring	A threat actor spoofs a legitimate account (user or service) and injects falsified data into the monitoring system to disrupt operations, create a diversion, or mask the attack.	MONITORING: falsified data injected/spoofing, tampering, Integrity, injects falsified data into the monitoring system to disrupt	P2-15 Aggregate, Store and Report on Product Collected Monitoring, Planning and Feedback Data	161 Infrastructure Manipulation	Advanced Persistent Threat Insider Threat Architect Cybersecurity Engineer	SC1 Mitigation Strategy 1	Keep at the Meta Level and better explained in the "star
3	Unauthorized Access/Modifies logs to divert attribution	A threat actor gains unauthorized access to logging data, alters system logs to conceal illicit activity from forensic audits, automated responses and alerts, or to divert attribution.	Logs: insider threat modifies the logs to conceal activity	P2-15 Aggregate, Store and Report on Product Collected Monitoring, Planning and Feedback Data	161 Infrastructure Manipulation	Insider Threat Site Reliability Engineer Cybersecurity Engineer		
4	Inadequately configures system logging	A threat actor has configured the collection of system logs in a way that limits the effectiveness of forensic audit activities.	Accidentally misconfiguring Logging - can't perform forensics work against what is captured	P2-15 Aggregate, Store and Report on Product Collected Monitoring, Planning and Feedback Data	176 Configuration/Environment Manipulation	Software Developer		Could be 1617 Most significa improper configuration
5	Intentionally misconfiguring	A threat actor has configured the collection of system logs in a way that limits the effectiveness of forensic audit activities in order to conceal subsequent activities.	Intentionally misconfiguring the system	P2-15 Aggregate, Store and Report on Product Collected Monitoring, Planning and Feedback Data	176 Configuration/Environment Manipulation	Insider Threat		
6	Intentionally locks out accounts responsible for recovering, investigating, or repairing the system	A threat actor spoofs an individual's account in order to create user action logs with the objective of making a targeted user in violation of security policy and reducing the targeted individual's organizational effectiveness.	Targeting individual with the intent that their login is denied, locking out individuals who should have access	P2-15 Aggregate, Store and Report on Product Collected Monitoring, Planning and Feedback Data	212 Functionality Misuse	Insider Threat		Could be a CAPEC - 184 So Attack
		Unit testing is insufficient to cover the requirements and abuse cases. A software or site reliability engineer doesn't		P2-15 Aggregate, Store and Report on Product Collected	176 Configuration/Environment	Software Developer		

Threats Link

Capturing the Complexity of the System



Example of Threats Traced to Capabilities via Operational Activities

Configuration Management Complexity Link

Summary

The goal of every program is to deliver a solution that is:

- Trustworthy – No exploitable vulnerabilities exist, either maliciously or unintentionally inserted.
- Predictable – When executed, software functions as intended and only as intended.
- Timely – Features are delivered as the speed of relevance

Security by design is achieved through integrating defensive thinking throughout the entire lifecycle.

Contact Information

Timothy A. Chick

Applied Systems Group Technical Manager, CMU-Software Engineering Institute

Adjunct Faculty Member, CMU-Software and Societal Systems Department

tchick@sei.cmu.edu

https://s3d.cmu.edu

https://www.sei.cmu.edu