Prepare for Increased Inherited Risk with DevSecOps in a High Assurance Context

Carol Woody, Ph.D.
Principal Researcher

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

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.

DM21-0050

Topics

Origins of Inherited Risk

Key Impacts of Inherited Risk for DevSecOps

Planning for Future Risk Must Begin Today

Prepare for Increased Inherited Risk with DevSecOps

Origins of Inherited Risk

Software is Everywhere

You think you're building (or buying, or using) a product such as:

car or truck	satellite	mobile phone	development tools
home security system	aircraft	pacemaker	security tools
home appliance	financial system	bullets for a gun	

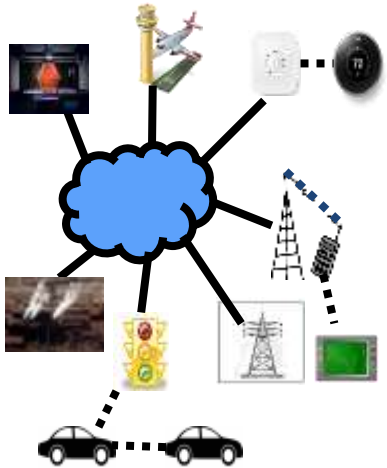
Actually you're getting ***a software platform:***

- Software is a part of almost everything we use.
- Software defines and delivers component and system communication.
- Software is used to build, analyze and secure software.

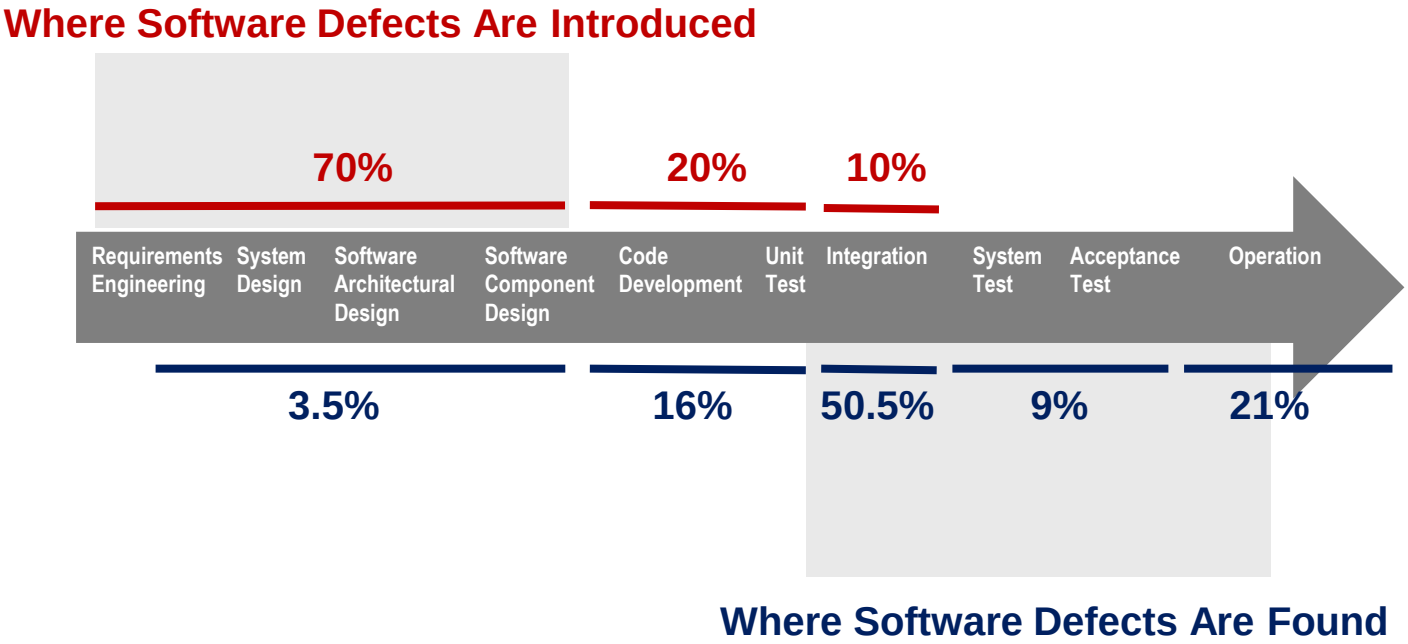
All software has defects:

- 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)



Most Software Defects Are Found Long After They Are Introduced



Sources: *Critical Code*; NIST, NASA, INCOSE, and Aircraft Industry Studies
**Woody et al. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=428589)*

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.

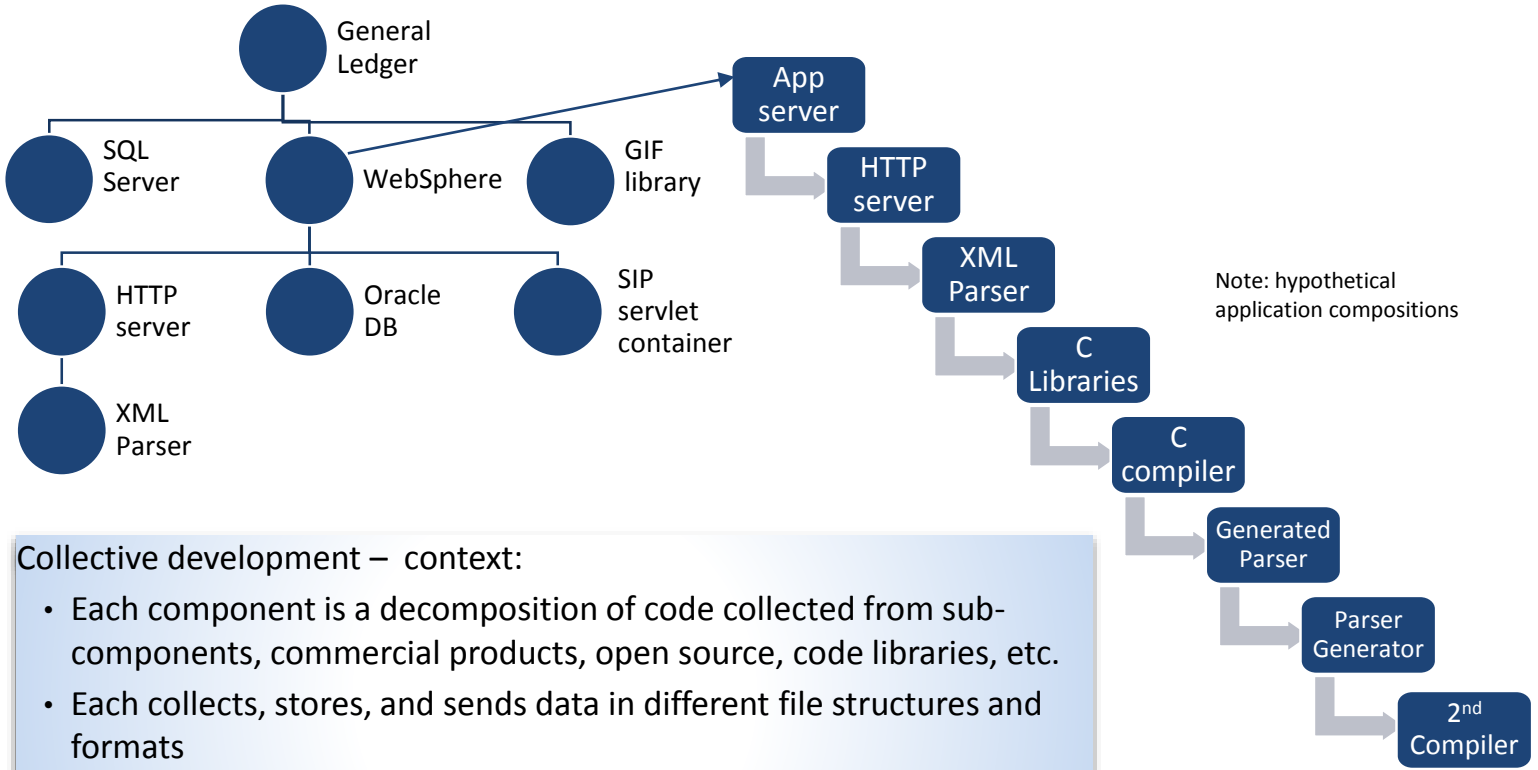
Access

- 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.

Software Development is Now Module Assembly



Note: hypothetical application compositions

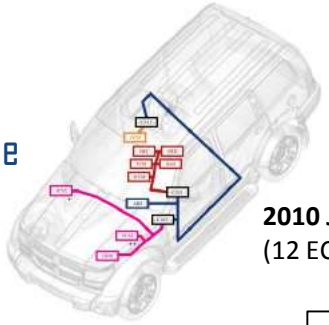
Collective development – context:

- Each component is a decomposition of code collected from sub-components, commercial products, open source, code libraries, etc.
- Each collects, stores, and sends data in different file structures and formats
- No one person, team, or organizations knows how all the pieces work

Reuse is rampant!

Modularity is Emphasized: Assemble from 3rd party components 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)

Expect, and engineer for, increases in supply chain risk

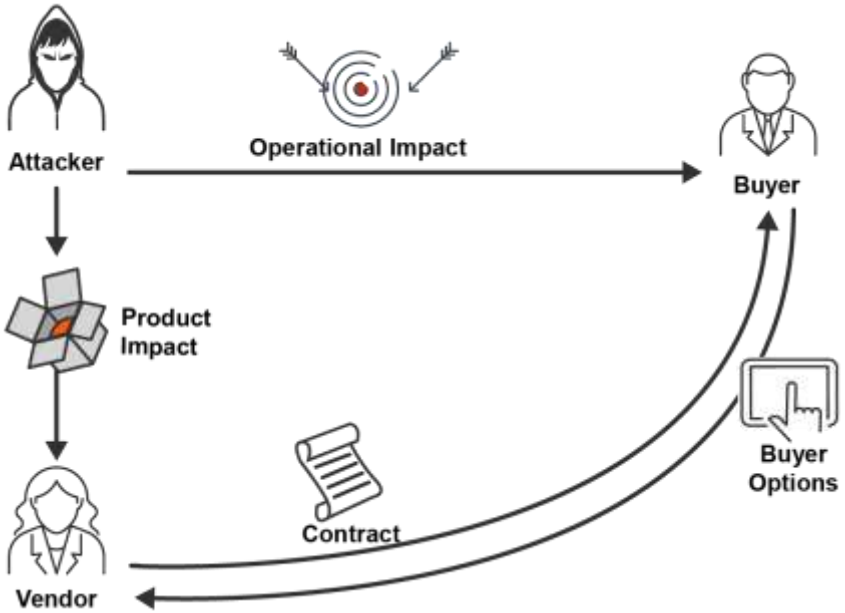
ECUs are prefabricated, software-driven components addressing select functionality and tailorable to a specific domain.

Modern high-end automotive vehicles have software and connectivity:

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

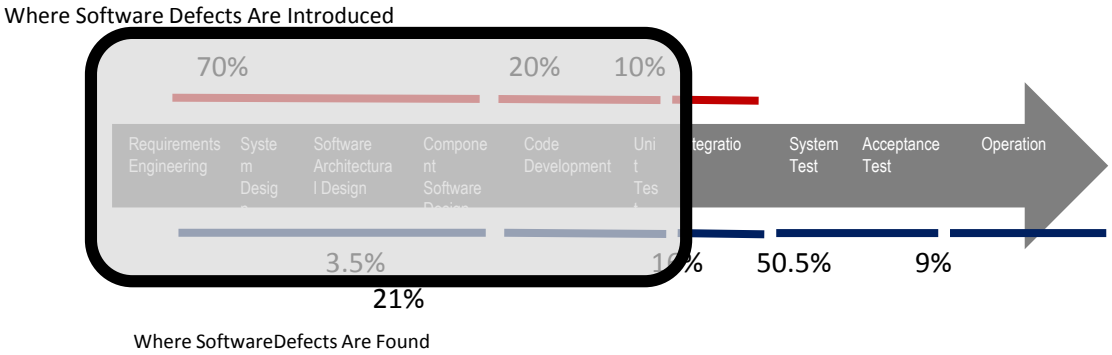
Sources: Miller and Valasek, A Survey of Remote Automotive Attack Surfaces, http://illmatics.com/remote%20attack%20surfaces.pdf; https://www.cst.com/webinar14-10-23~?utm_source=rfg&utm_medium=web&utm_content=mobile&utm_campaign=2014series
https://en.wikipedia.org/wiki/Electronic_control_unit

Components of a Supply Chain Relationship



Acquirer Visibility is Limited Unless Visibility is Contracted

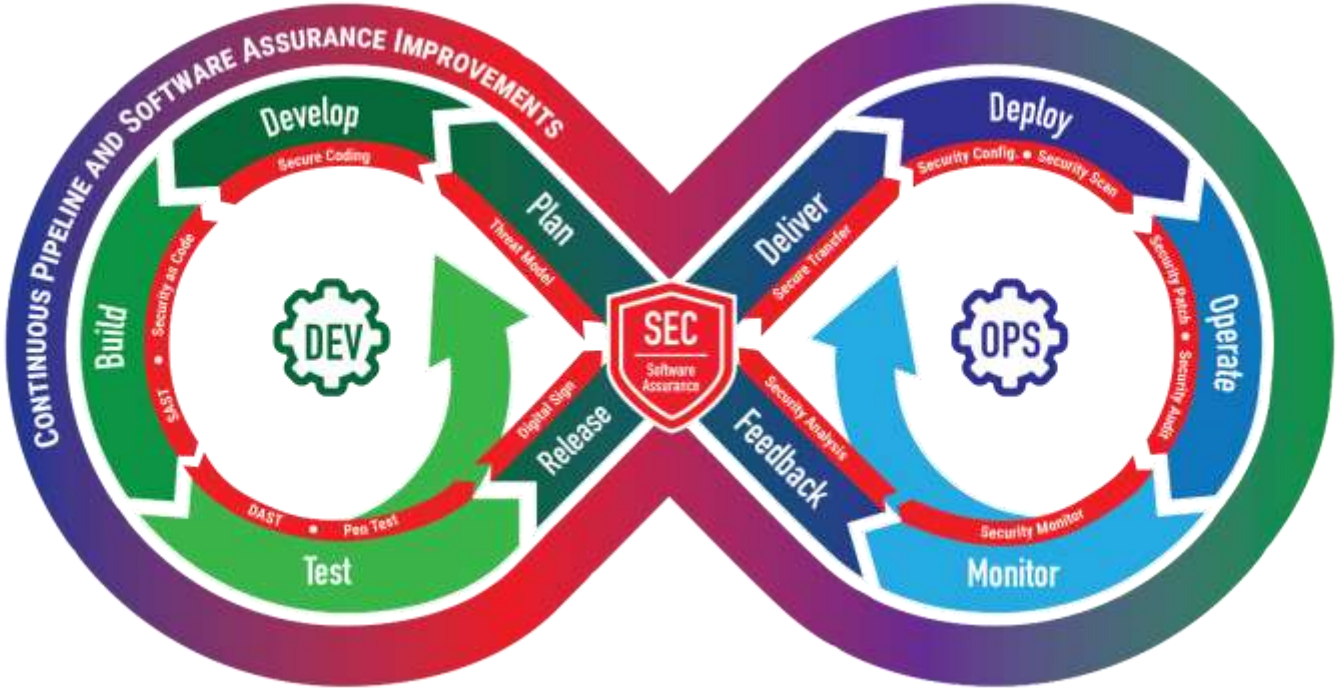
With supply chains, monitoring is indirect
Improvements require a focus on SCRM activities early in the acquisition to establish vendor processes and practices that reduce defects



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

Key Impacts of Inherited Risk for DevSecOps

DevSecOps Integrates Many 3rd Party Components Into the Pipeline



Criticality of Configuration Management (CM) Increases

Infrastructure as Code (IaC) and related Security as Code (SaC) capabilities extend CM to **all** configuration files, tests scripts, source code, property files, binaries, servers, tools, log files, etc.

- Some are housed in the cloud repositories further complicating how they are controlled and who has access
- Read access to such information would provide a blueprint to the entire system in order to analyze and identify weaknesses.
- Write access would allow unauthorized changes to the system to be automatically propagated through the development pipeline and into production.

Cloud Acquisition Decisions Will Impact All Lifecycle Phases

Cloud contracts can severely limit what tools, data, access, and capacity are available for testing and operations

- Current onsite processes assume full physical access and maximum operational capacity is always available
- Access to data about the operational environment must come from the Cloud Service Provider (CSP) and be negotiated as part of the contractual arrangements

Quality of service (availability, latency, and throughput) will be a major issue as Cloud connections increase

- Acceptable levels need to be established for the program with the CSP and enforced for all steps in the lifecycle
- Program influence on these decisions will depend on the type of acquisition

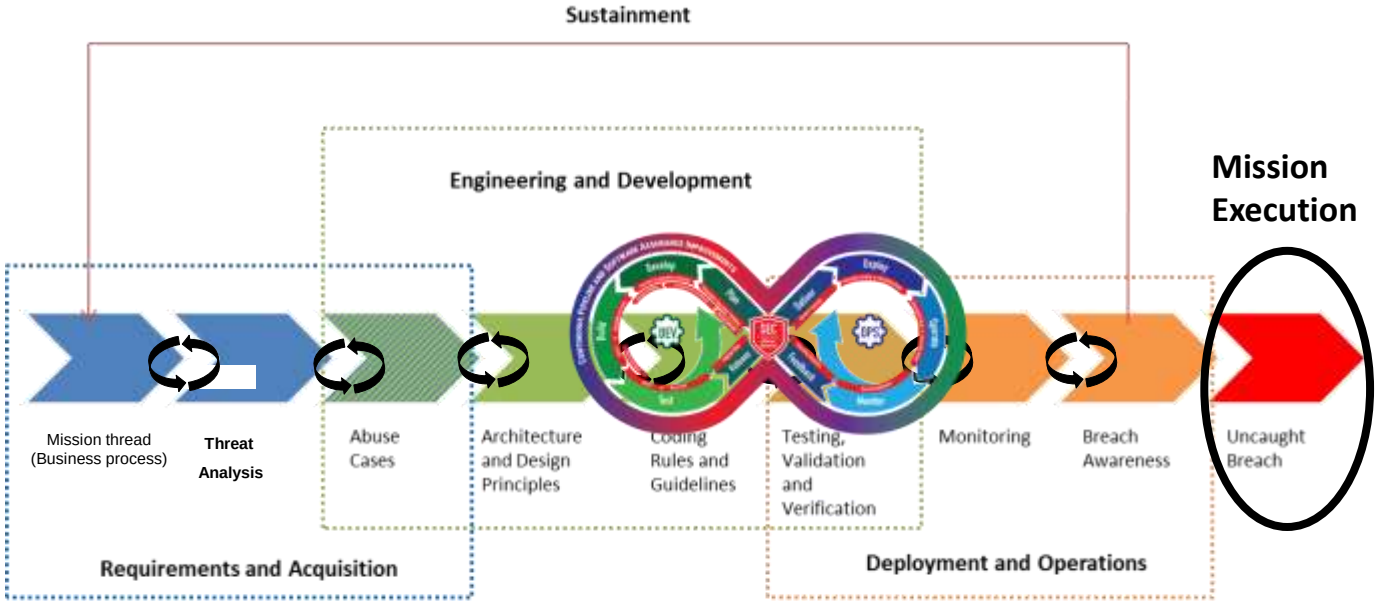
Acquirer practices using available tools will establish the risk of data in the Cloud; DevSecOps administrators will determine who has access to these tools

Planning is Critical to Risk Management

Comprehensive CM can support automated notification of unexpected changes, verification of all test and script executions, and up to date manifests for all software packages, components, and libraries, including associated risks:

- This requires a means to differentiate normal from abnormal activities.
- Logs will be generated from many tools and processes which will need to be joined for analysis using roles of normality.
- This information provides a map of what is and is not monitored and parameters that define normal behaviors
  - extremely valuable to an adversary in order to avoid detection
  - managed by administrative resources that grant access who probably know little about the pipeline

Cybersecurity Risk Is a Lifecycle Challenge and the Pipeline is Only One Piece



Prepare for Increased Inherited Risk with DevSecOps

Planning for Future Risk Must Begin Today

Chasing Software Flaws is a Chronic Activity

The National Institute of Standards and Technology (NIST) National Vulnerability Database (NVD) contains **152,766 known vulnerabilities** – NVD received **15,911 new vulnerabilities in 2020** (as of 11/9). Just a few items from **SANS NewsBites** (published Tuesdays & Fridays) and **SANS @Risk** (published Thursdays) https://www.sans.org/newsletters/ (a few of hundreds from 14 August through 12 November 2020)

- Microsoft Patch Tuesday updates address at least 120 vulnerabilities in Windows and other products and services, including two actively exploited vulnerabilities
- Universal Health Services (UHS) Ransomware Attack Affects All 400 U.S. Health Systems
- Improperly Configured AWS S3 Bucket Exposes 10 Million Hotel Guest Records
- Google Drive Collaboration Feature is Being Exploited by Bad Actors
- Oracle WebLogic Server Unauthenticated Remote Code Execution Vulnerability

The pipeline must keep the tools patched and look for ways of improving them to find the latest flaws

Tools Move in and Out of Favor and Support

1979 – Chroot for container-style isolation added to Unix

2000's – Expansion of container technology for Linux and Solaris

2013 – Docker offered graphical user interface; Google started development of Kubernetes

2014 – Docker 1.0 downloaded 2.75 million times

2015 – Kubernetes released; Google partnered with Linux Foundation to form Cloud Native Computing foundation; Docker reorganized and sold Docker Enterprise and its orchestration tool (Docker Swarm) went on 2-year end of life

2019 – Kubernetes is the primary tool of choice

Pipeline and Product are Never Complete

Requirements

Architectural Modeling & Design

Prediction

Continuous Development

Operational Fielding

Integrated Testing & Certifications

Verification

Tools & Flow Control Development Platform

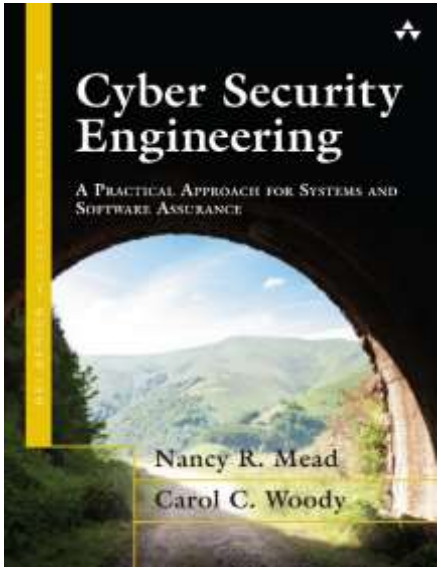
Are you funded and staffed for continuous change and improvement?

Carnegie Mellon University
Software Engineering Institute

Opportunities to Learn More About Cybersecurity Engineering

Textbook (SEI Book Series)

Cybersecurity Engineering



Professional Certificate

CERT Cybersecurity Engineering and Software Assurance

https://sei.cmu.edu/education-outreach/credentials/credential.cfm?customel_datapageid_14047=33881

Online training in five components

- Software Assurance Methods in Support of Cybersecurity Engineering
- Security Quality Requirements (SQUARE)
- Security Risk Analysis (SERA)
- Supply Chain Risk Management
- Advanced Threat Modeling

Contact Information

Carol Woody, Ph.D.

cwoody@cert.org

Web Resources

Building security into application lifecycles

https://sei.cmu.edu/research-capabilities/all-work/display.cfm?customel_datapageid_4050=48574

CMU SEI Home Page

https://sei.cmu.edu/