TRUSTED CI

THE NSF CYBERSECURITY
CENTER OF EXCELLENCE

trustedci.org

Software Engineering for NSF Science and CI

Susan Sons

IU-CACR, TrustedCI

sesons@iu.edu

TRUSTED CI

THE NSF CYBERSECURITY CENTER OF EXCELLENCE

Overview

- Landscape
- Software Development/Security Programs
- MVP (Minimally Viable Program)
- Level 2

---- BREAK ----

- Level 3
- Level 4
- Building and Maturing Your Program
  (aka story time)
- Q&A

The Landscape:

What are science's real software challenges?

Science is Different.

Goals

- Reproducibility
- Integrity
- Sustainability

Constraints

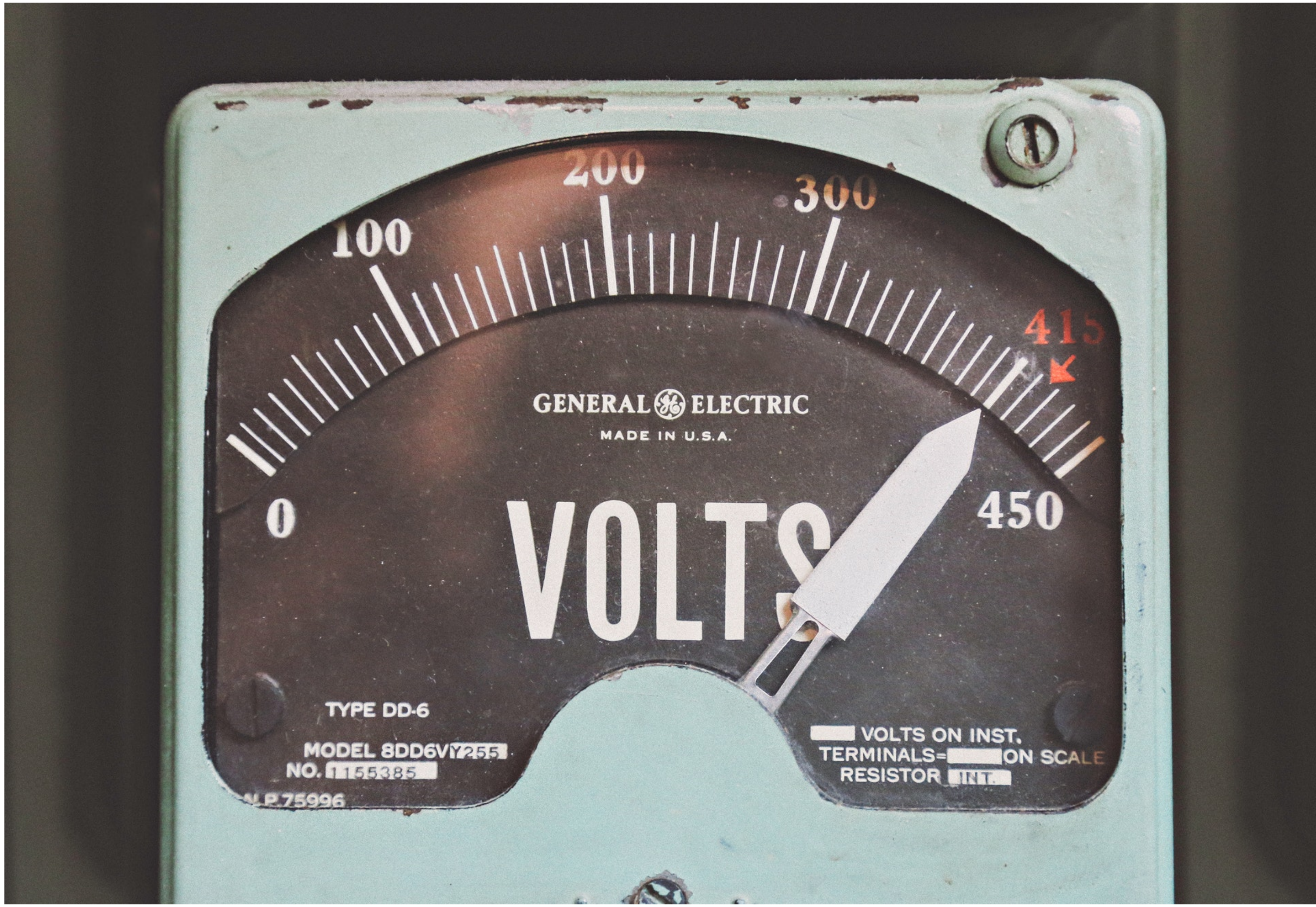
- Life Cycle
- Accidental Developers
- Time travel effect

Software Development and Security Programs

What makes it a program?

- Ongoing activity
- Budget
- Goals
- Iteration
- Fault Tolerance

How much is enough?



Four Factors

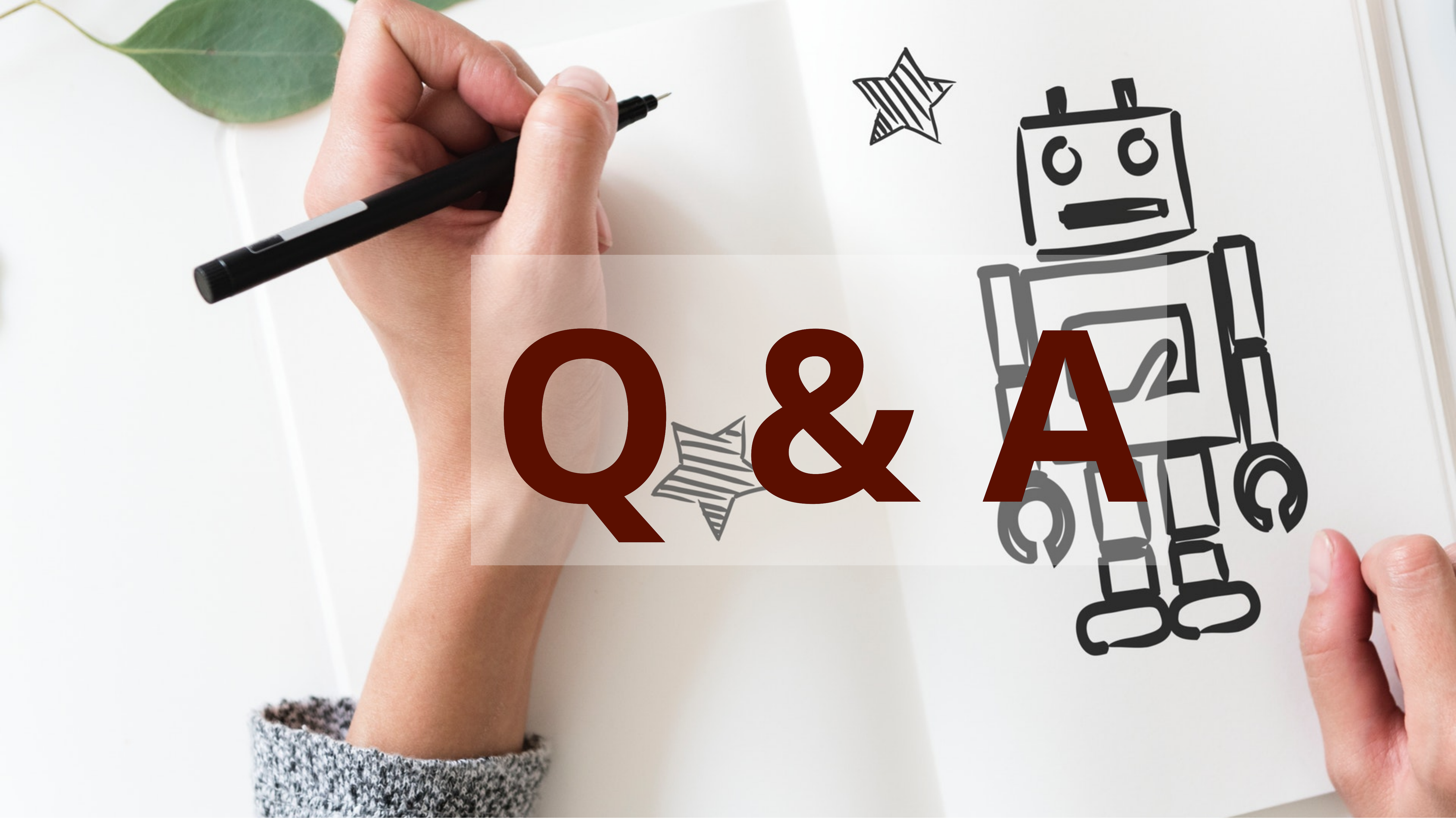
- Policy: ad hoc or formalized
- Communication (internal and external)
- Resources, tools, and expertise
- Consistency

Six Levels:

0. Do nothing.

1. MVP: recommended for small one-off experiments and POCs.
2. Basic SWE Practice: for non-CI that is shared.
3. Default Level: for widely used science software and most CI.
4. For high-reliability CI.
5. For critical CI and code with high assurance requirements.

Q & A



MVP: Minimally Viable (software engineering) Program

MVP Goals:

- **Integrity:**
  Reviewers know exactly what software touched the research data. CI owners can trace problems when the software runs.
- **Reproducibility:**
  Peers can examine the source, or build and re-use the software, days or decades later in order to attempt to reproduce the results.
- **Continuing the Scientific Process:**
  Software without a license is presumed to be "all rights reserved". Other researchers cannot touch it.
- **Clarity:**
  Those considering use of the software should know what it does and what state it's in.

MVP Features:

- Revision Control
- Documentation:
  dependencies and build process
- Build System
- Changelog
- Development Status
- License

Revision Control

Documentation

Build System

Changelog

Development Status

License

Q & A

Six Levels:

0. Do nothing.

1. MVP: recommended for small one-off experiments and POCs.
2. Basic SWE Practice: for non-CI that is shared.
3. Default Level: for widely used science software and most CI.
4. For high-reliability CI.
5. For critical CI and code with high assurance requirements.

Level 2: Basic Software Engineering Practice

Start with MVP Features:

- Revision Control
- Documentation:
  dependencies and build process
- Build System
- Changelog
- Development Status
- License

Level 2 adds:

- Revision Control Usage Patterns
- Semantic Versioning
- Distribution Planning
- Code Signing
- Basic Security Policy, to include Vulnerability Management
- Dependency Selection
- Succession Planning
- Issue Tracker
- Testing

Revision Control Usage Patterns

Branching, Tagging, and Authoritative Repository

Semantic Versioning

https://semver.org

Software Distribution Plan

Channels, Notifications, Frequency

Code Signing

Dependencies

Succession

Issue Tracker

Testing

Q & A

TRUSTED CI

THE NSF CYBERSECURITY
CENTER OF EXCELLENCE

trustedci.org

Six Levels:

0. Do nothing.

1. MVP: recommended for small one-off experiments and POCs.
2. Basic SWE Practice: for non-CI that is shared.
3. Default Level: for widely used science software and most CI.
4. For high-reliability CI.
5. For critical CI and code with high assurance requirements.

Level 3: Default Level

Review

Level 1

- Revision Control
- Documentation:
  dependencies and build process
- Build System
- Changelog
- Development Status
- License

Level 2

- Revision Control Usage Patterns
- Semantic Versioning
- Distribution Planning
- Code Signing
- Dependency Selection
- Succession Planning
- Issue Tracker
- Testing

Level 3 Adds:

- Least Privilege and Code Review
- Basic Security Policy, to include Vulnerability Management
- Coding Standards
- Automated and Manual Testing Reqs
- Automated Builds
- Development Documentation
- Issue Tracker Management
- Up/Down Stream Communication
- Architectural Review
- Security Exercises

Least Privilege and Code Review

Coding Standards

Software Security Policy

Must include vulnerability management.

Bare Minimum Security Policy:

NTPSec, circa 2016

"The NTPSec Project Manager, Mark Atwood, accepts risk on behalf of the NTPSec project. The NTPSec Information Security Officer, Susan Sons, has the authority to declare an incident and direct its remediation."

Software Security Policy Considerations:

- Who accepts risk?
  NOT your security officer.
- Who leads an incident?
- What happens if one of these people is unavailable?
- What standards do we follow on a daily basis?
- Who can make policy exceptions, and how?
- What documentation is done, and when is it reviewed?
- When is the policy reviewed/updated?
- How are inside and outside vulnerability reports handled?
- What disaster plans do we have?

Testing:
Automated and Manual

Automated Builds

Development Documentation

Issue Tracker Management

Up/Down Stream Communication

Architectural Review

Security Exercises

Q & A

Six Levels:

0. Do nothing.
1. MVP: recommended for small one-off experiments and POCs.
2. Basic SWE Practice: for non-CI that is shared.
3. Default Level: for widely used science software and most CI.
4. For high-reliability CI.
5. For critical CI and code with high assurance requirements.

Six Levels:

0. Do nothing.

1. MVP: recommended for small one-off experiments and POCs.
2. Basic SWE Practice: for non-CI that is shared.
3. Default Level: for widely used science software and most CI.
4. For high-reliability CI.
5. For critical CI and code with high assurance requirements.

Level 1

- Revision Control
- Documentation: dependencies & build process
- Build System
- Changelog
- Development Status
- License

Level 2

- Revision Control Usage Patterns
- Semantic Versioning
- Distribution Planning
- Code Signing
- Dependency Selection
- Succession Planning
- Issue Tracker
- Testing

Level 3

- Least Privilege, Code Review
- Basic Security Policy
  - Vulnerability Management
  - Coding Standards
- Automated and Manual Testing Reqs
- Automated Builds
- Development Docs
- Issue Tracker Management
- Up/Down Stream Comms
- Architectural Review
- Security Exercises

Level 4 Adds:

- Next iteration on policy
- Static analysis, when available
- Expectations management
- Formal change review process
- Maturing the security exercise program
- Release cycle management
- Understanding downstream to ultimate deployments

Iterating on Policy

Static Analysis

Expectations Management

Formal Change Review

Maturing the Security Exercise Program

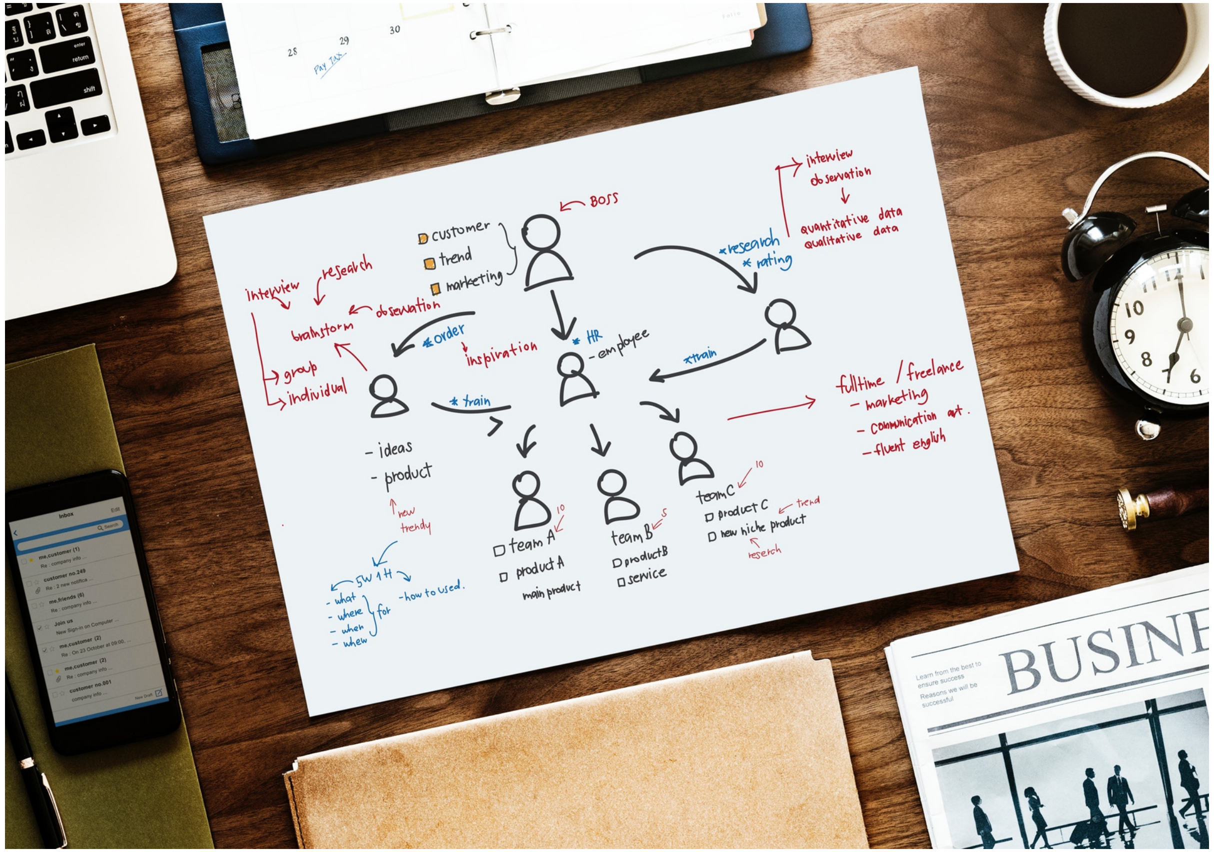
Release Cycle Management

Understanding Downstream to Ultimate Deployments

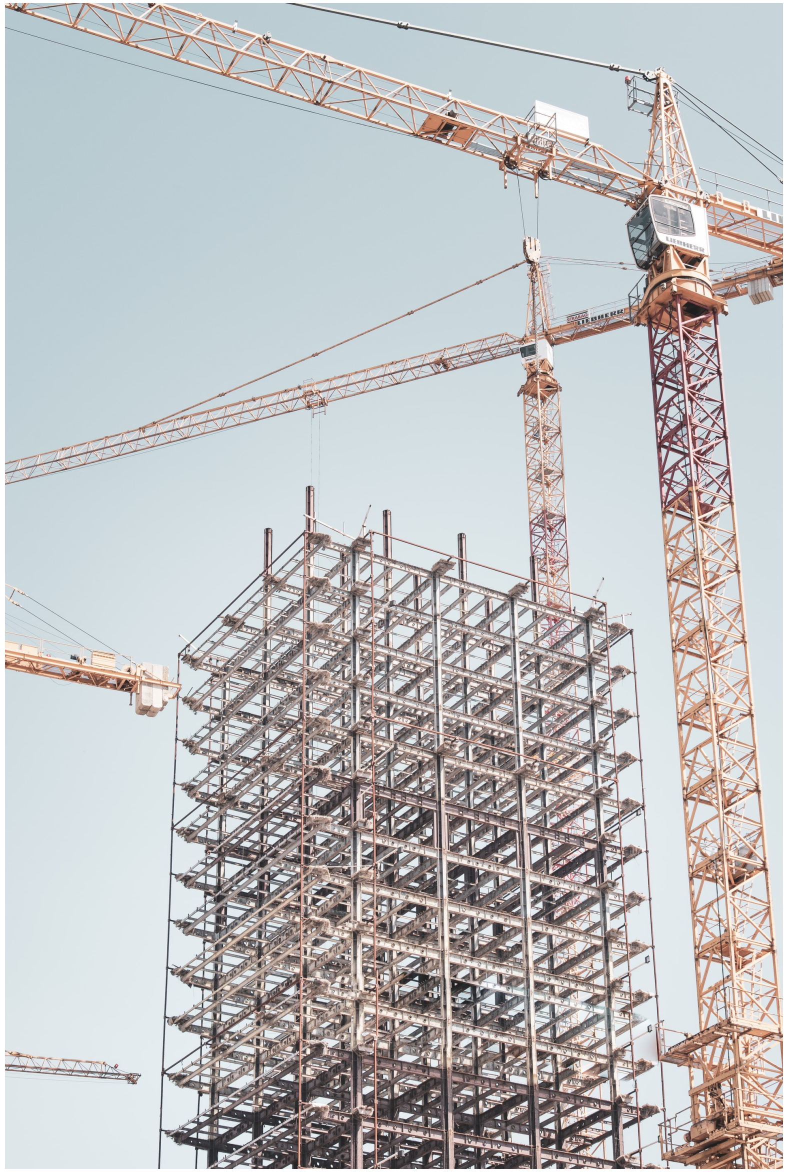
Q & A

In the end, it's about adoption...

Best Case: Introducing the Security Program at Conceptualization



Second-Best Case: Introducing the Security Program During the Initial Build Phase



Strategies for phasing security into an active development project.

Improving what you've got

Q & A

Thank you for your time and attention!

Here are some resources to continue your journey:

- TrustedCI: https://trustedci.org
- IU CACR: https://cacr.iu.edu
- Talk notes and slides will go up at https://security.engineering/talks within 48 hours.

sesons@iu.edu

sons@security.engineering

Using and Sharing This Work:

Please credit Susan Sons and the IU Center for Applied Cybersecurity Research when using this presentation.

Permissions beyond the scope of this license may be available; send inquiries to sons@security.engineering.

The most current version of this presentation is available from

https://slides.com/hedgemage/nsf-summit18-swe-guide