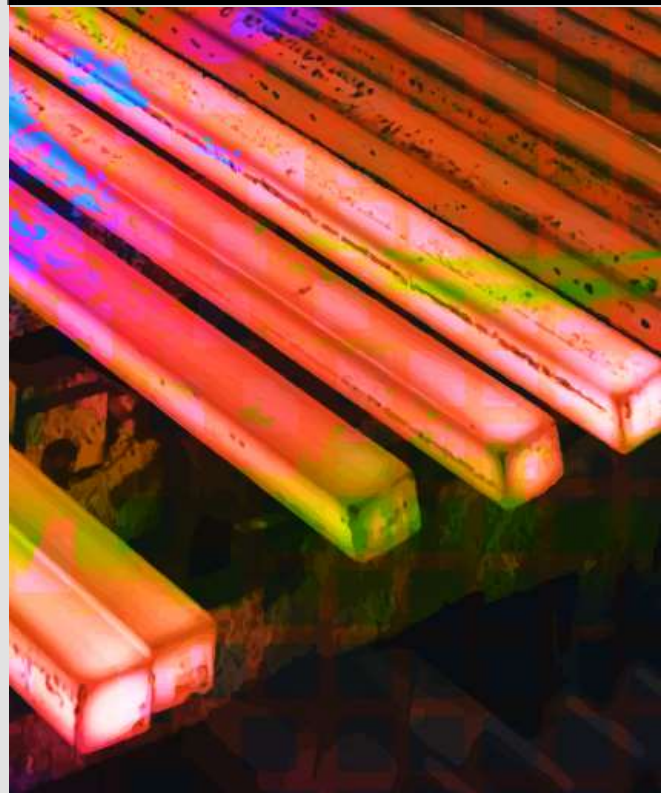
Carnegie Mellon University
Software Engineering Institute

Realities of SBOM: What is Under the Hood of SBOM



MAY 11, 2023

Hasan Yasar
Technical Director and Faculty Member

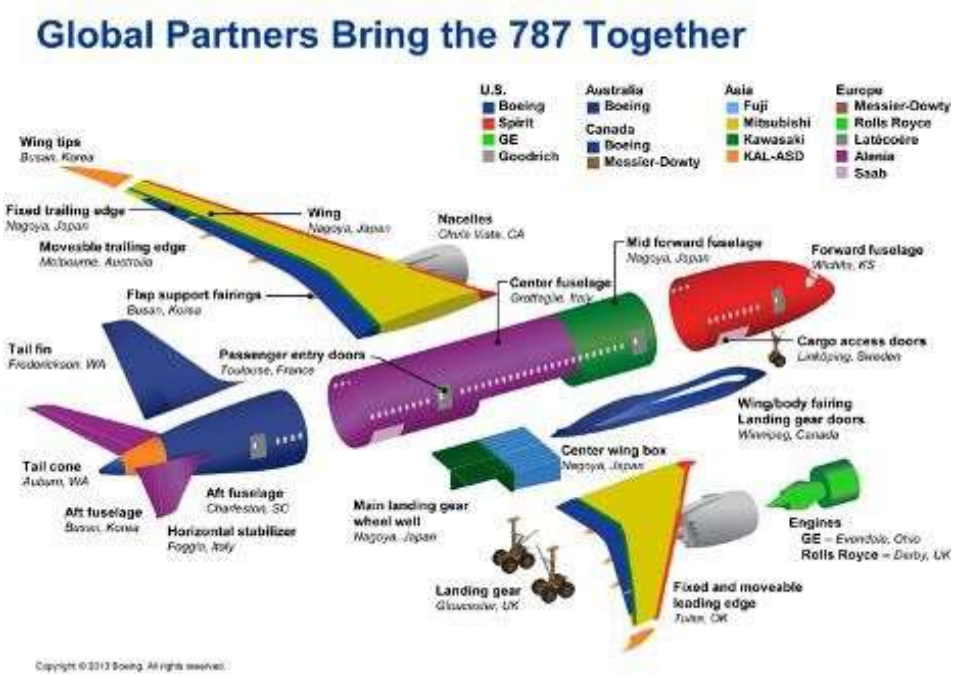
This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM23-0478

Yesterday's Enterprise

Source: www.wikipedia.org

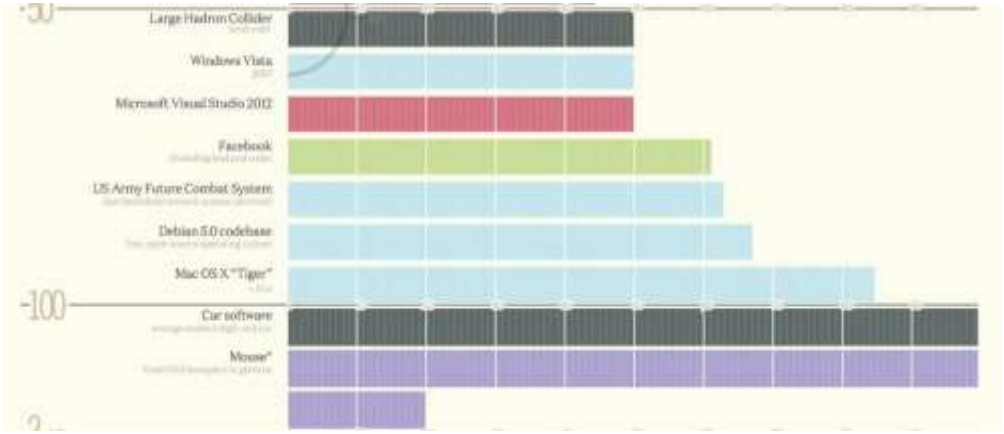
Growing Supply Chain Complexity



Software is Everywhere

"In short, software is eating the world." - Marc Andreesen

Source: www.wjs.com



Source:https://informationisbeautiful.net/visualizations/million-lines-of-code/

DEPARTMENT OF TRANSPORTATION
Federal Aviation Administration 14 CFR Part 39

[4910-13-P]

[Docket No. FAA-2015-0936; Directorate Identifier 2015-NM-058-AD; Amendment 39-18153; AD 2015-09-07]
RIN 2120-AA64
Airworthiness Directives; The Boeing Company Airplanes **AGENCY:** Federal Aviation Administration (FAA), DOT. **ACTION:** Final rule; request for comments.
SUMMARY: We are adopting a new airworthiness directive (AD) for all The Boeing Company Model 787 airplanes. This AD requires a repetitive maintenance task for electrical power deactivation on Model 787 airplanes. This AD was prompted by the determination that a Model 787 airplane that has been powered continuously for 248 days can lose all alternating current (AC) electrical power due to the generator control units (GCUs) simultaneously going into failsafe mode. **This condition is caused by a software counter internal to the GCUs that will overflow after 248 days of continuous power.** We are issuing this AD to prevent loss of all AC electrical power, which could result in loss of control of the airplane.

Source: www.edgardaily.org

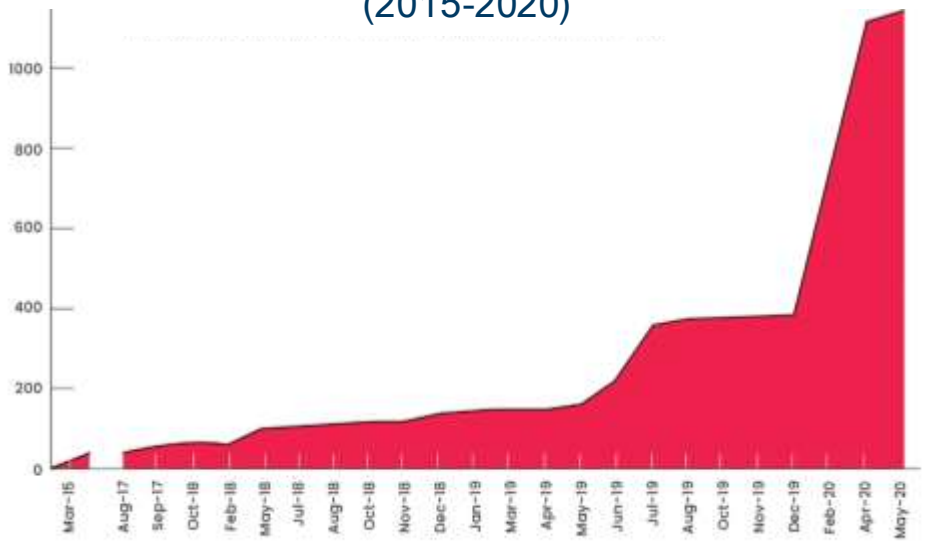
Why does it matter?

97% of commercial code contains at least some open source codes[1]

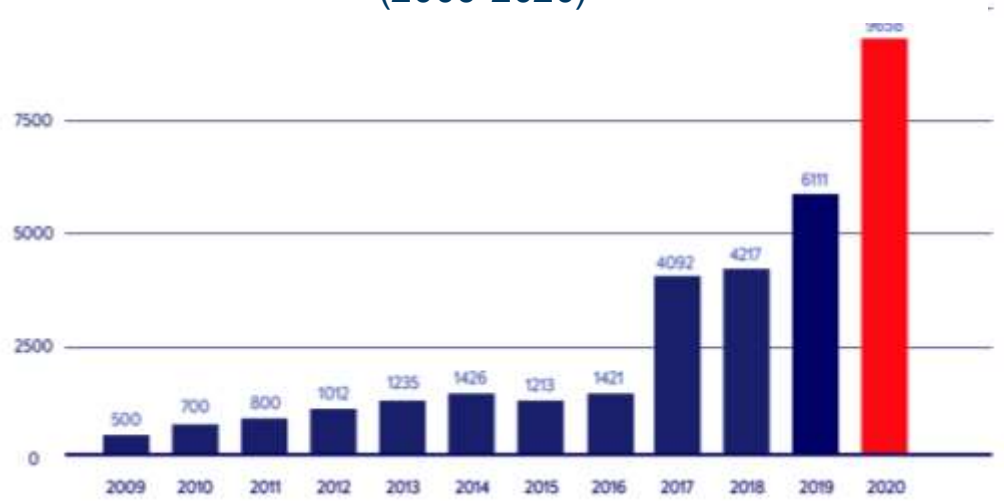
81% of codebases contain an outdated version of open source[2]

62% of breaches originated from a compromised software component[3]

Software Supply Chain Attacks[4]
(2015-2020)



Open Source Vulns per Year[5]
(2009-2020)



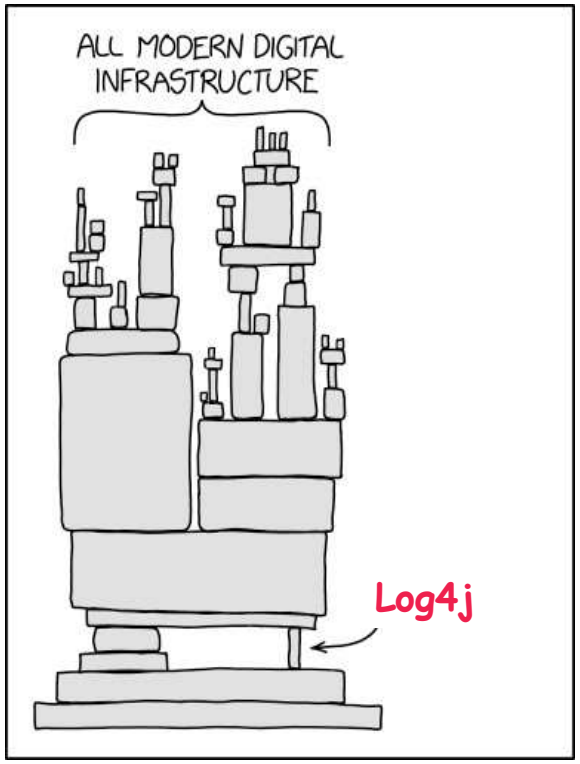
[4] *Sonatype Software Supply Chain Attack report 2020*
[5] *Mend Annual Report, Open Source Vulnerabilities 2021*

[1+2] *Synopsys OSSRA report 2022*
[3] *Verizon Data Breach Investigations Report 2022*

Why does it matter? (continued)

When breached, a single exploitable software component can compromise countless services.

Use Case: Log4j – a software component embedded in Java-based products and web services.



- *Exemplary ROI*
- *≈10,000 person-hours* with an estimated ad-hoc response cost equal to **$400K – $900K**

- A breach incident could have resulted in an average financial risk range of **$141K – $5.7M**[5]

Targeting Software Development

Adversaries target software development (source, build, and deployment systems): SolarWinds, ASUS Live/Shadowhammer, MEDoc/Not Petya, others.

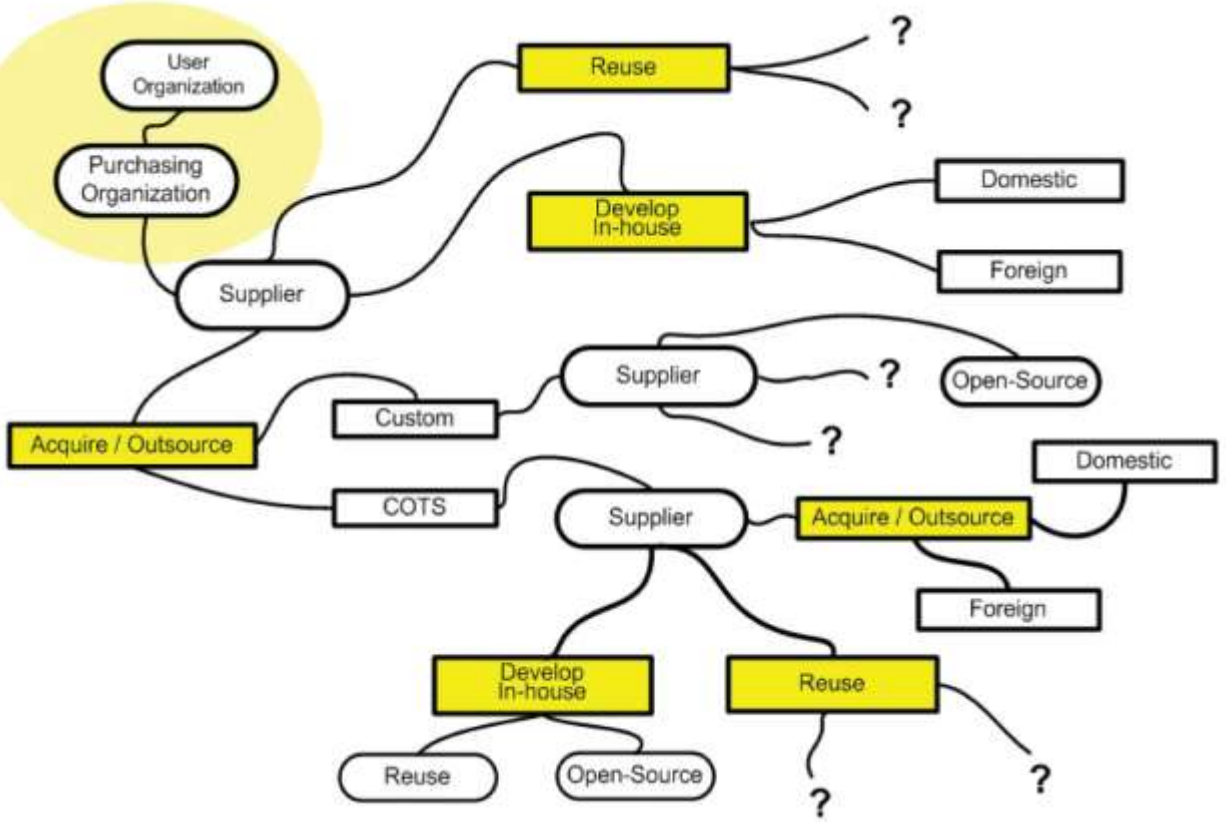
Reflections on Trusting Trust, Ken Thompson, 1984:

> "No amount of source-level verification or scrutiny will protect you from using untrusted code. In demonstrating the possibility of this kind of attack, I picked on the C compiler. I could have picked on any program-handling program such as an assembler, a loader, or even hardware microcode. As the level of program gets lower, these bugs will be harder and harder to detect. A well-installed microcode bug will be almost impossible to detect."[18]

https://www.cs.cmu.edu/~rdriley/487/papers/Thompson_1984_ReflectionsonTrustingTrust.pdf

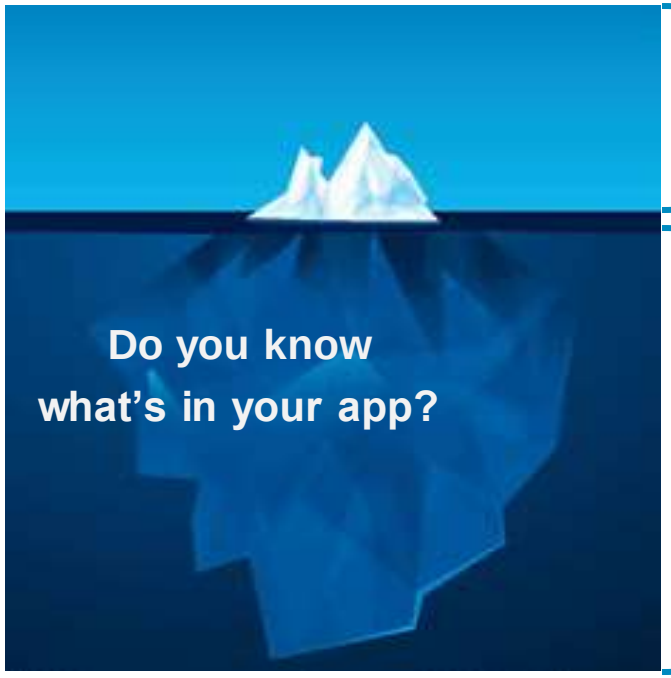
Software supply chain for assembled software

- Expanding the scope and complexity of acquisition and deployment
- Visibility and direct controls are limited (only in shaded area)



Source: "Scope of Supplier Expansion and Foreign Involvement" graphic in DACS www.softwaretechnews.com Secure Software Engineering, July 2005 article "Software Development Security: A Risk Management Perspective" synopsis of May 2004 GAO-04-678 report "Defense Acquisition: Knowledge of Software Suppliers Needed to Manage Risks"

But SW Development became like a College Party



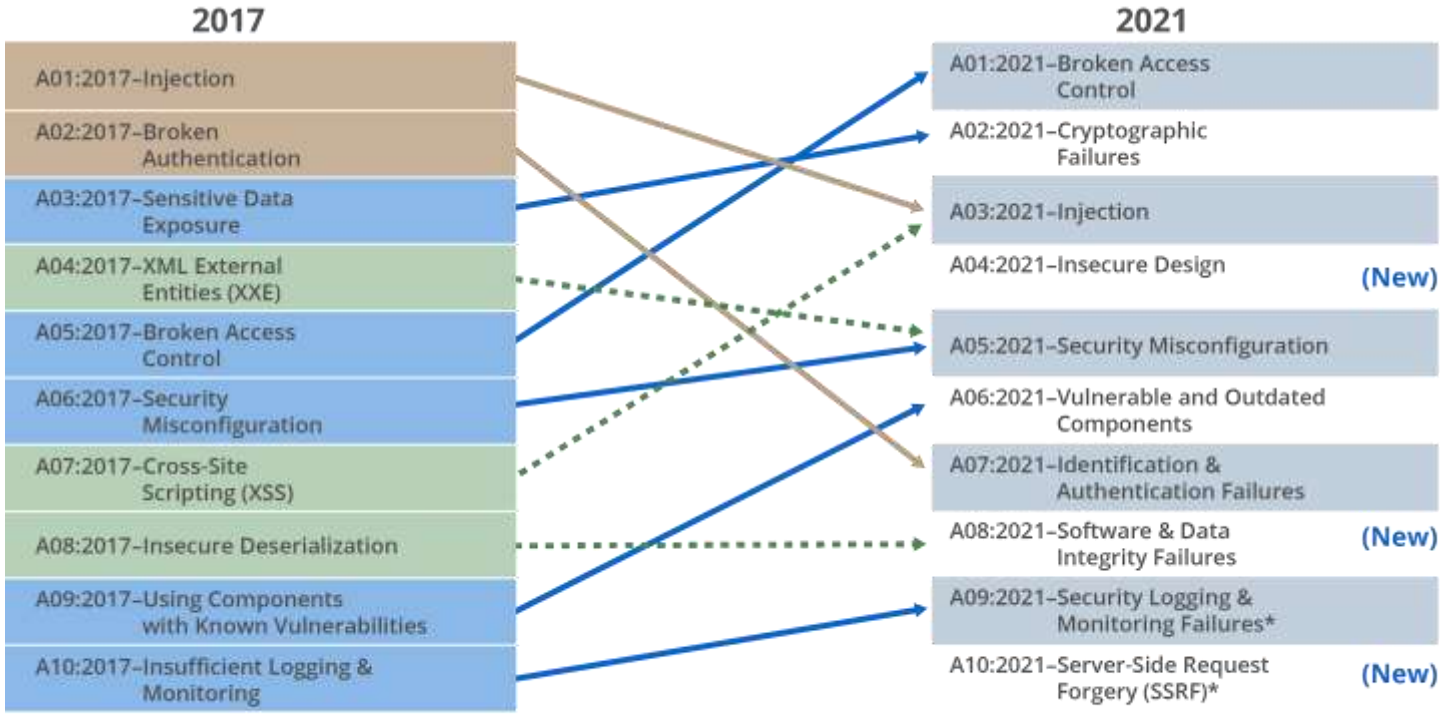
Code we wrote

Code someone else wrote

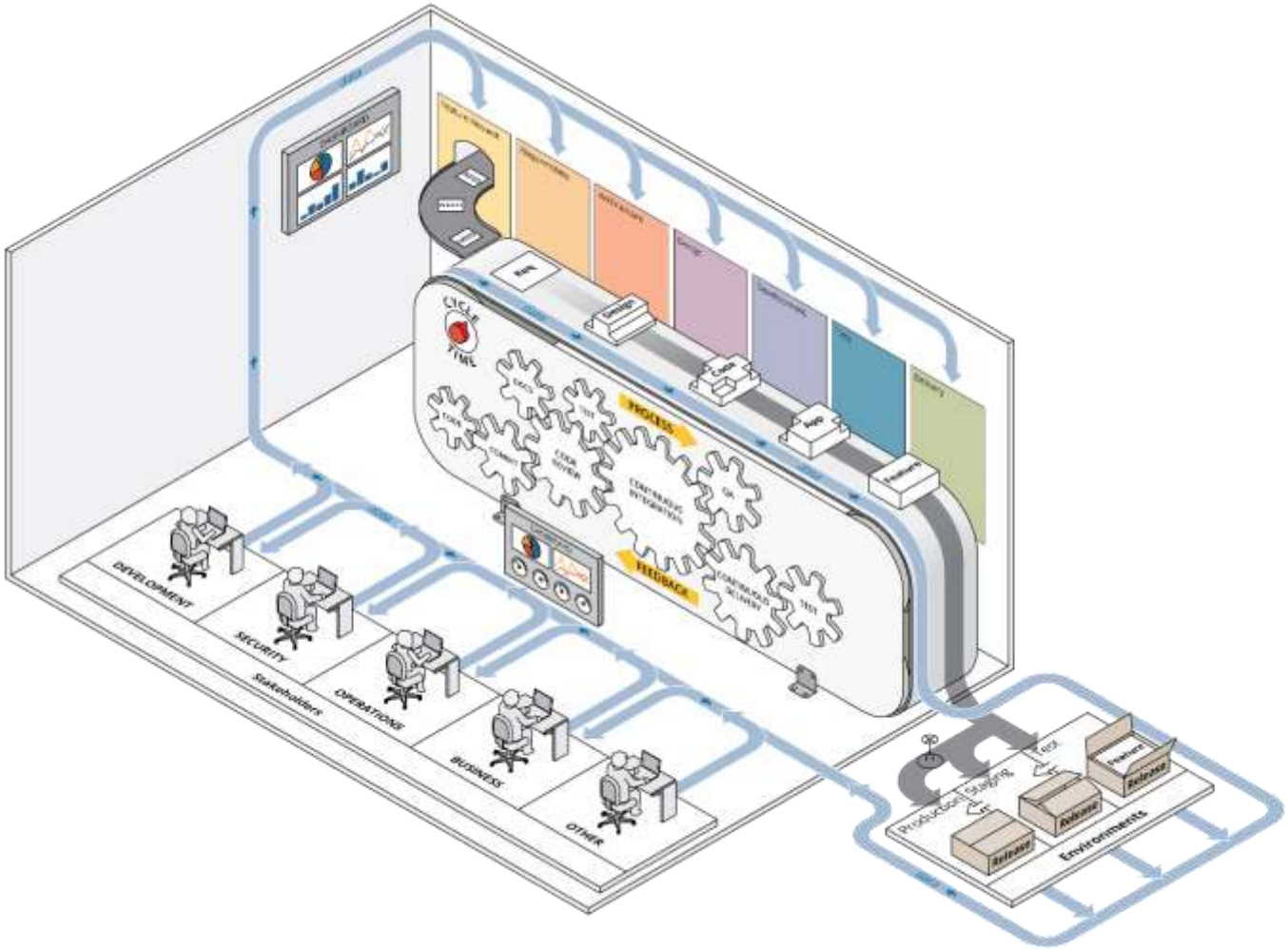
Is SBOM is all about creating with tools?

Our Goal: Secure Software!

Smarter software requires safer and more secure **design**, **development**, and **deployment** into **secure** infrastructures.



DevSecOps

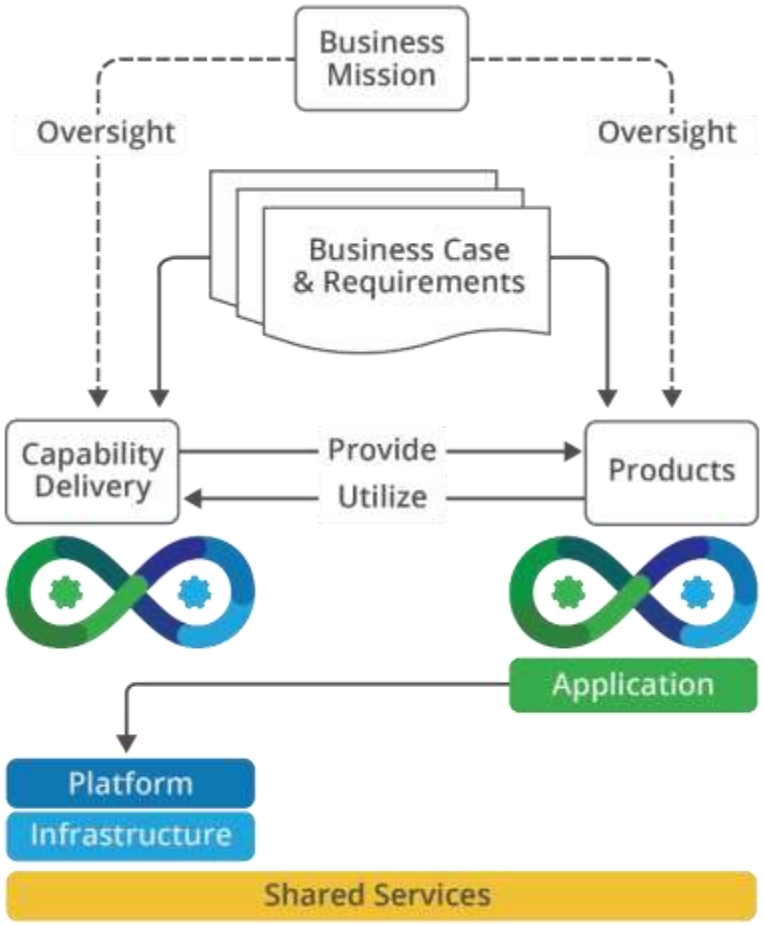


- "DevOps is a set of principles and practices which enable better communication and collaboration between relevant stakeholders for the purpose of specifying, developing, continuously improving, and operating software and systems products and services." [1]

- "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." [2]

[1] IEEE 2675 *DevOps Standard for Building Reliable and Secure Systems Including Application Build, Package and Deployment*
[2] *DevSecOps Guide: Standard DevSecOps Platform Framework*. U.S. General Services Administration. https://tech.gsa.gov/guides/dev_sec_ops_guide.

DevSecOps Goal

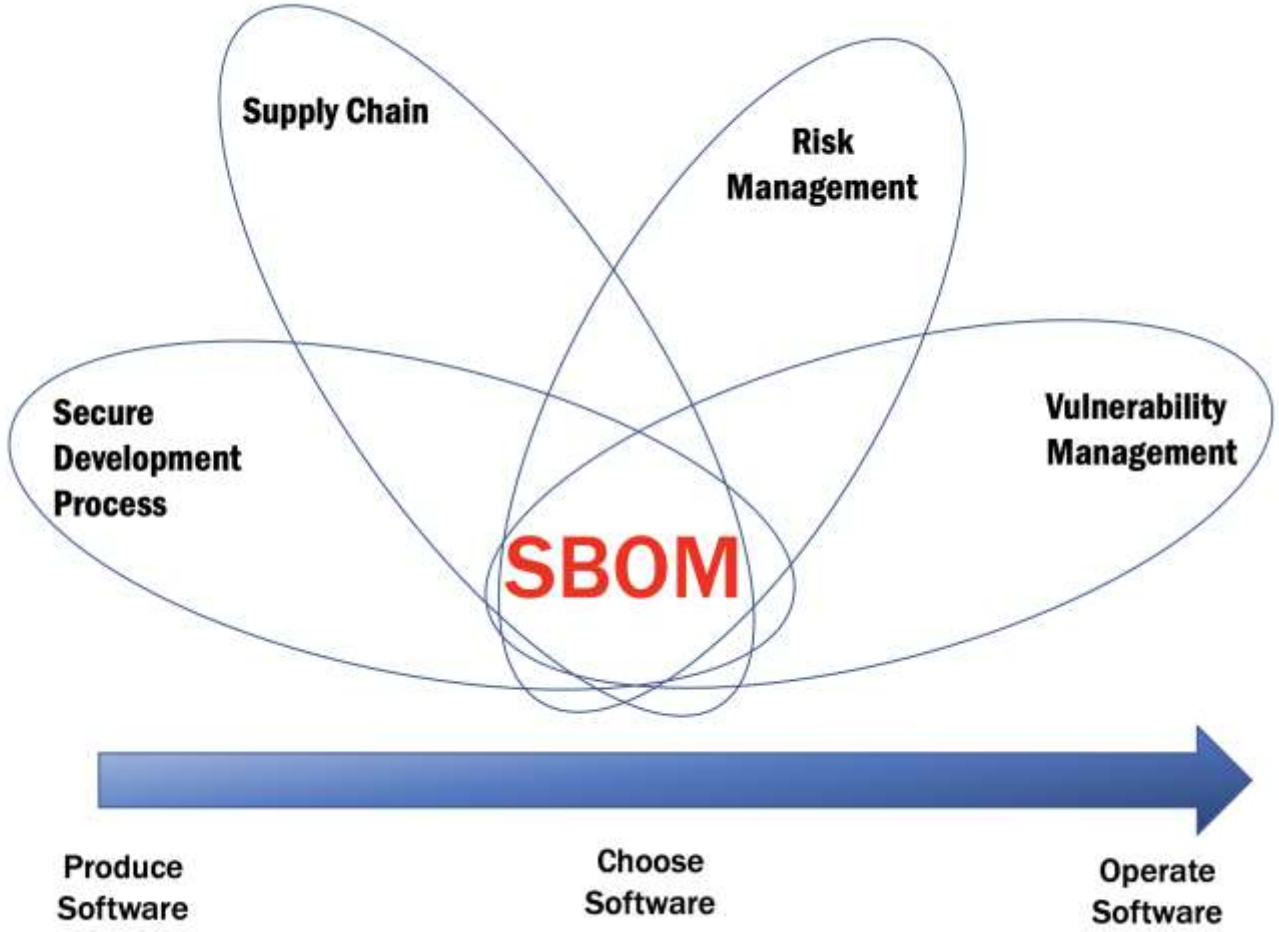


DevSecOps-oriented enterprises are driven by three concerns:

- *Business Mission* captures stakeholder needs and channels the whole enterprise to meet those needs. It answers the questions: *Why does the enterprise exist?* and *For Whom does the enterprise exist?*
- *Capability to Deliver Value* covers the people, processes, and technology necessary to build, deploy, and operate the enterprise's products.
- *Products* are the units of value delivered by the organization. Products utilize the capabilities delivered by the software factory and operational environments.

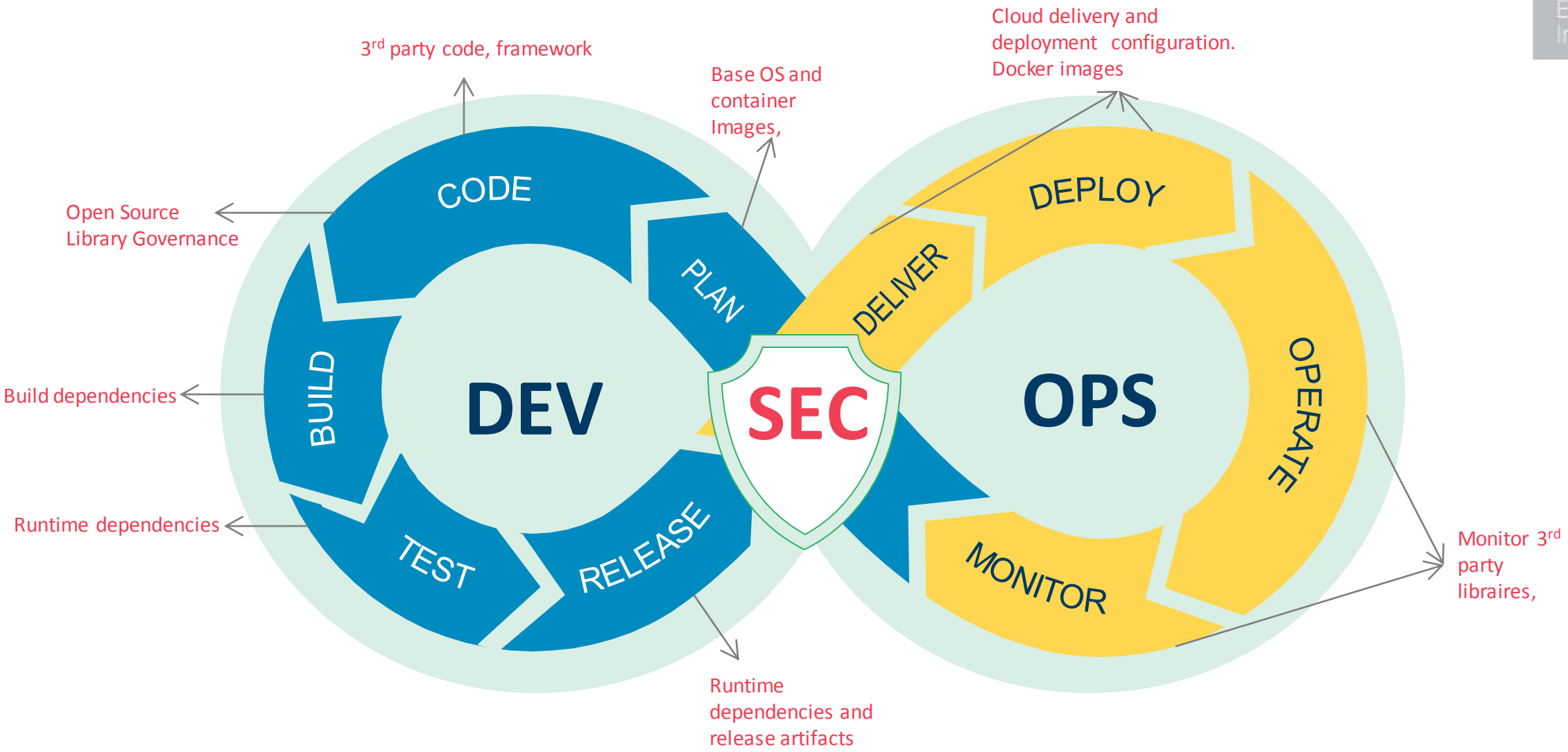
SBOM and DevSecOps

- SBOM should be integrated into SDLC across DevSecOps practices and process
- SBOM should ne integrated with Risk and Vulnerability malmanagement
- SBOM will be used to response any new security findings for libraires or code under application stack

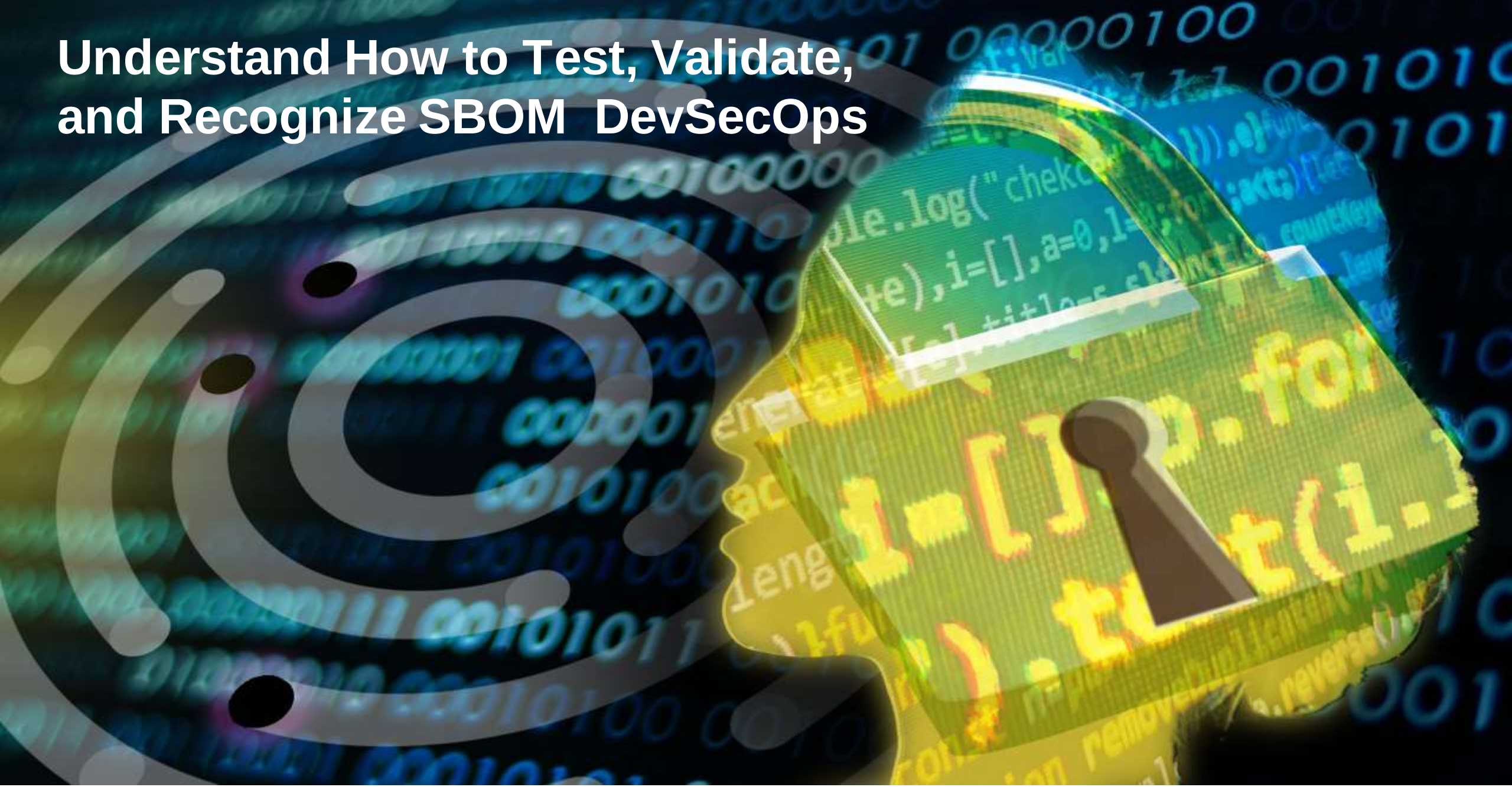


15

SBOM across DevSecOps



Understand How to Test, Validate, and Recognize SBOM DevSecOps



Realities of SBOM

But How to Start?

Carnegie Mellon University Software Engineering Institute

Common Format for SBOM

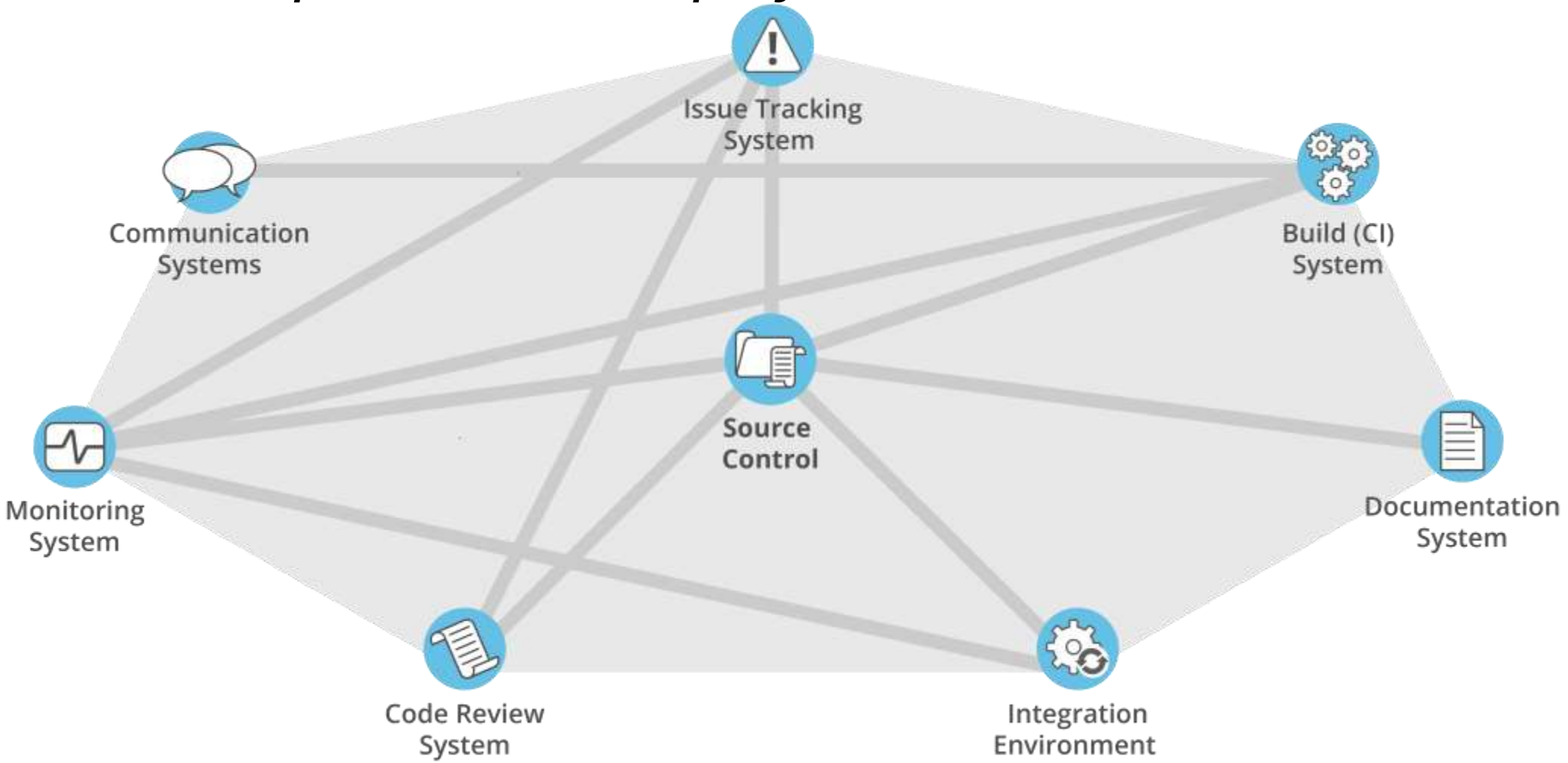
SPDX is an open standard for communicating software bill of material information (including components, licenses, copyrights, and security references). The SPDX specification is developed by the SPDX workgroup, which is hosted by The Linux Foundation. The grass- roots effort includes representatives from more than 20 organizations— software, systems and tool vendors, foundations and systems integrators.

CycloneDX is a software bill of materials (SBOM) standard, purpose-built for software security contexts and supply chain component analysis. The specification is maintained by the CycloneDX Core working group, with origins in the OWASP community

SWID tags record unique information about an installed software application, including its name, edition, version, whether it is part of a bundle and more. SWID tags support software inventory and asset management initiatives. The structure of SWID tags is specified in international standard ISO/IEC 19770-2:2015.

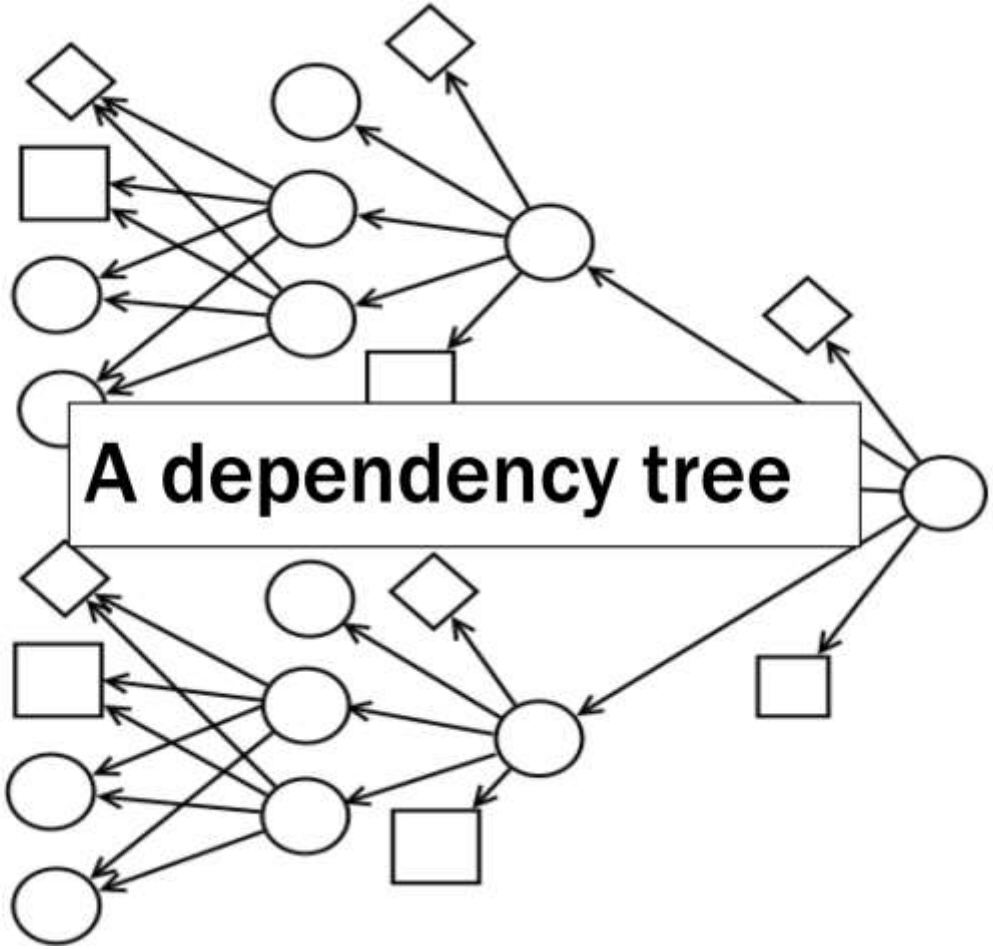
- All has common elements.
- A 'multilingual' ecosystem does not offer too many challenges
- Rather than pick a winner, develop guidance to support all formats with effective interoperability.

Start to Build a Secure pipeline for your *Development and Deployment*



Then use secure pipeline for your app!

- Create and maintain software dependency for each build
- Track ALL 3rd party, including open-source libraries, used in i) code development, ii) build, and iii) runtime process



Implementing core SBOM fields

Elements	CycloneDX
Supplier	publisher
Component	name
Unique Identifier	bom/serialNumber and component/bom-ref
Version	version
Component Hash	hash
Relationship	(Nested assembly/subassembly and/or dependency graphs)
SBOM Author	bom-descriptor: metadata/manufacture / contact

CycloneDX uses a Package URL to uniquely identify a version of a dependency and its place within an ecosystem. It looks like this:

```
pkg:maven/org.jboss.resteasy/resteasy-jaxrs@3.1.0-Final?type=jar
```

The *Package URL* identifies all relevant component metadata, including ecosystem (type), group (namespace), name, version, and key/value pair qualifiers.

Operations – How Does It All Fit

SBOM Policy (Design/Source)

1

Policy to acquire/enforce (Vendor SBOM)

- Establish Policy to mandates, software delivered to the enterprise from a supplier to include a contractual software bill of materials in common format
- Ask requestor to develop first set of SBOM files

2

SRA (OWASP Dependency Track)

- Perform impact analysis of supply chain software libraries.
- perform impact analysis on vendor SBOM using OWASP dependency track
- Use analysis output as an additional input to overall risk assessment quantification.

CI/CD Policy Compliance (Build, Analyze, Deploy)

1

Continuous Integration (GitLab)

- Build pipelines in GitLab submit finished artifacts to artifactory for deployment phase activities.
- Developers will have visibility to component composition and vulnerabilities through GitLab

2

Policy Compliance (Artifactory)

- VG to create and maintain policies that identify security violations based on matching conditions of CVE severity.<New>
- Artifactory scans finished component artifacts and generates SBOM containing CVE details.
- VG to measure policy violations across the portfolio, and against individual projects and components for SecureUP reporting. <New>
- Components failed from security policy scans are reported back to GitLab for remediation.

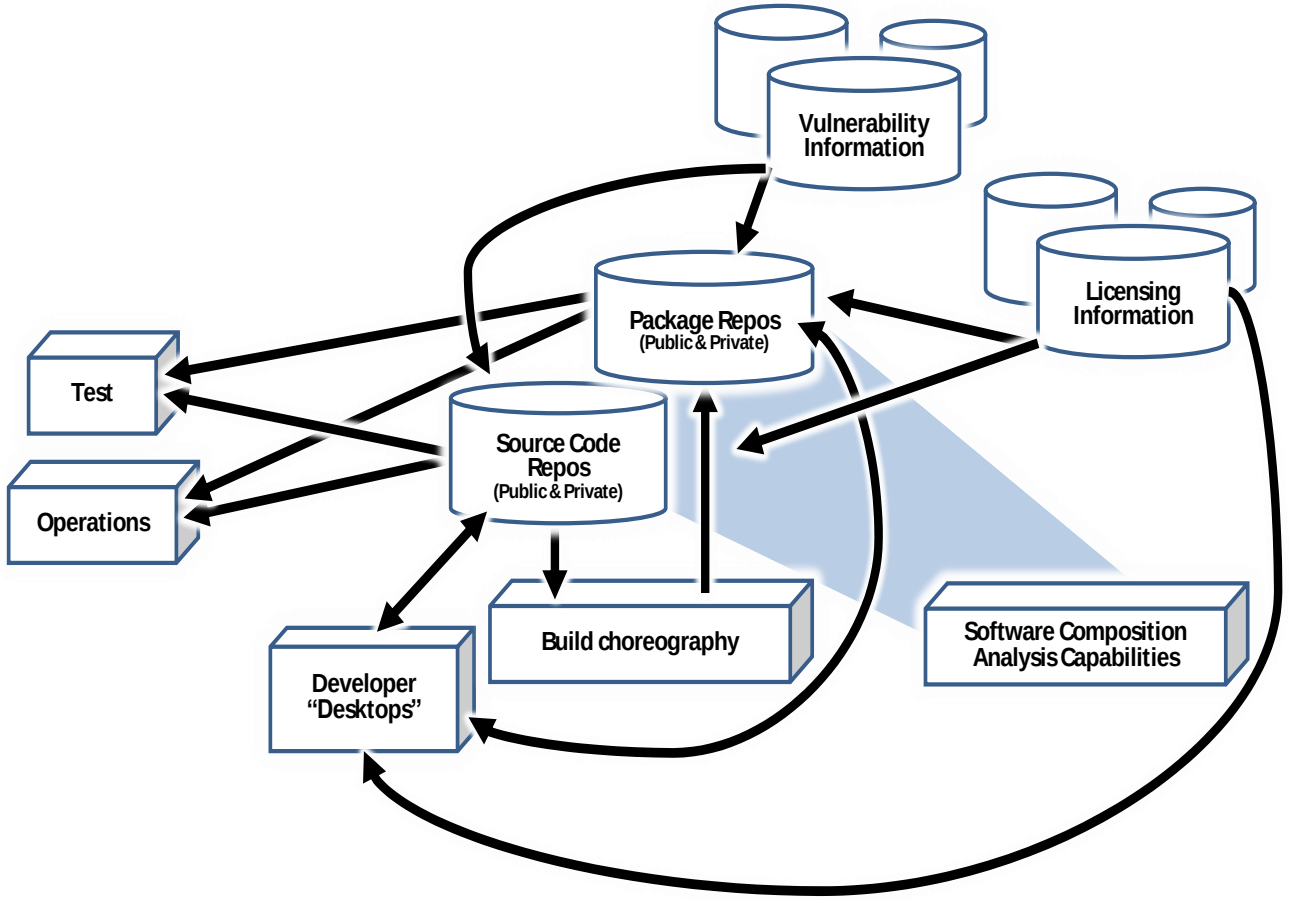
3

Continuous Deployment (Artifactory)

- Components that pass security and compliance policies check make to deployment.<New>
- Artifactory enforces security violation policy rules be blocking components getting into continuous deployment stage.

Integrate and automate SBOM

- Develop and implement artifact catalog
- Document automation scripts for dependencies
- Integrate SBOM files /scripts into build files
- Enable automated artifact pull/push one each build
- Develop automated scripts including artifacts and environment configuration
- Develop and update scripts to release any dependencies along with new version of the app
- Enable base containers/internal repository
- Continuously monitor and analyze dependencies
  - Security vetting process for each approved libraires
  - Monitor for each newly CVEs



Realities of SBOM

SBOM in action

Carnegie Mellon University
Software Engineering Institute

What is the project

- **Rust code base that contains a chain vulnerability**
- **Determine issues from supply chain dependencies**
- **Verify the process for building the project**
- **Execute checks of the code being written**
- **Execute checks of the external libraries**

Pipeline

- Static code analysis
- Build the code
- Package up support library
- Generate an SBOM(Software Bill Of Materials)
- Upload build artifacts
- Scan SBOM
- Verify the build commands

Demo Time

Carnegie Mellon University Software Engineering Institute

Let us see exemplary project to for SBOM lifecycle

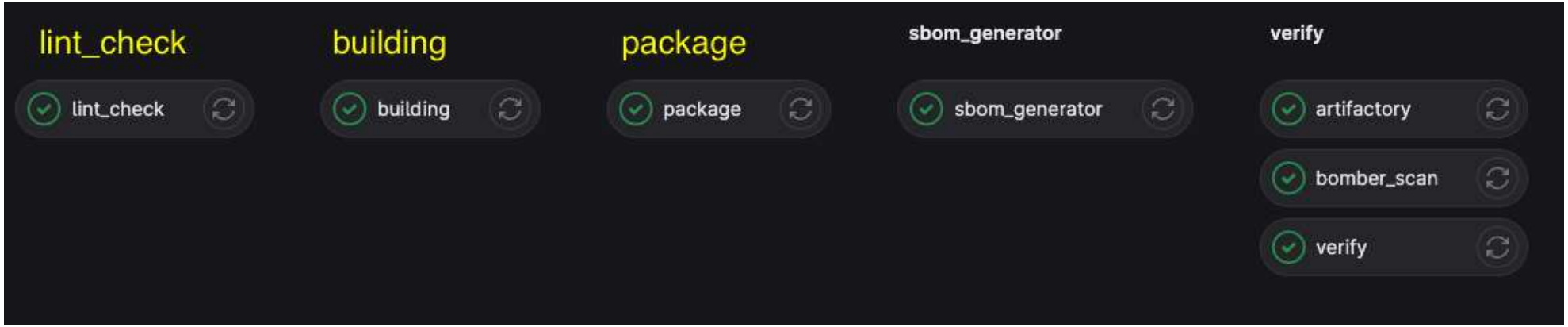
Realities of SBOM

Pipeline Stages

Carnegie Mellon University Software Engineering Institute

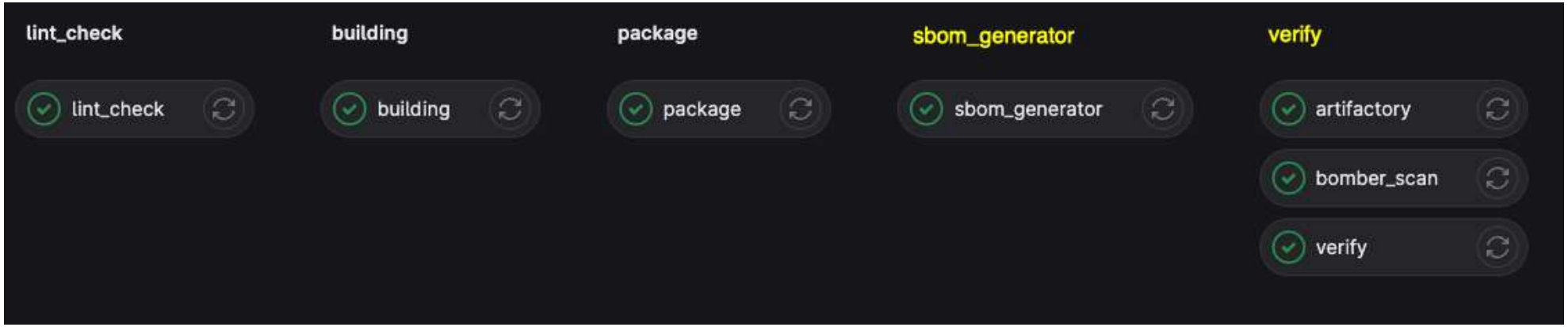
Lint, Build, and Package Stages

- Cargo Clippy
  - Static code analysis
  - Lint checks
- Cargo Build
  - Builds the crate
  - Builds the binary
  - Fails only if code errors
- Cargo Publish
  - Push the crate to Artifactory



SBOM and Verify Stages

- Software Bill Of Materials
  - CycloneDX
  - XML
- Artifactory
  - Push binary
  - Add build meta-data
- bomber_scan
  - Kungfu-Bomber
    - Vulnerability Scan SBOM
    - Check dependent crates
- verify
  - In-toto-verify the build steps



lint_check

$ in-toto-run -s -n lint_check -k ./.secure_files/gitlab -v -- cargo clippy

…

Checking supplychain-sandbox-wrap v0.1.9 (/var/lib/runner/builds/sei/supplychain-sandbox/supplychain-sandbox-wrap)

warning: variable does not need to be mutable

--> supplychain-sandbox-wrap/src/supplychain_sandbox_wrap.rs:17:9

|

17 | let mut v = vec![1, 2, 3];

| ----^

| |

| help: remove this `mut`

|

= note: `#[warn(unused_mut)]` on by default

arning: `supplychain-sandbox-wrap` (lib) generated 1 warning

Checking supplychain-sandbox v0.1.1 (/var/lib/runner/builds/sei/supplychain-sandbox)

Finished dev [unoptimized + debuginfo] target(s) in 23.17s

bom.xml

```
<bom …>
...
<component type="library" bom-ref="pkg:cargo/regex@1.5.4">
<name>regex</name>
<version>1.5.4</version>
<description>…</description>
<scope>required</scope>
...
<purl>pkg:cargo/regex@1.5.4</purl>
<externalReferences>
...</externalReferences>
</component>
...
</bom>
```

Bomber scanning

Carnegie Mellon University Software Engineering Institute

$ bomber scan bom.xml

DKFM - DevOps Kung Fu Mafia

https://github.com/devops-kung-fu/bomber

Version: 0.4.4

- Ecosystems detected: cargo
- Scanning 5 packages for vulnerabilities...
- Vulnerability Provider: OSV Vulnerability Database (https://osv.dev)
- Files Scanned

bom.xml (sha256:dadcdb030f572f30441136ea1088242a33d7bf213ff74b9265e83c587d05485f)

- Licenses Found: Unlicense OR MIT, MIT OR Apache-2.0

TYPE	NAME	VERSION	SEVERITY	VULNERABILITY	EPSS %
cargo	regex	1.5.4	UNSPECIFIED	CVE-2022-24713	52%
		1.5.4	HIGH	CVE-2022-24713	52%

Realities of SBOM

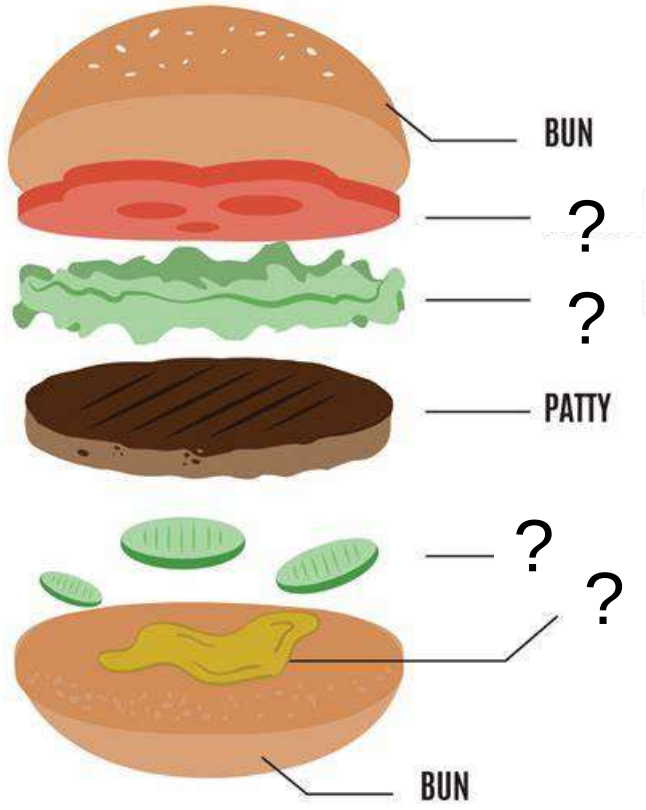
What is next?

Carnegie Mellon University
Software Engineering Institute

How do we solve it?

Software Bill of Materials (SBOM) needs to be considered mandatory for any effective DevSecOps or AppSec effort.

Do I always want to know what's in my food before eating it?

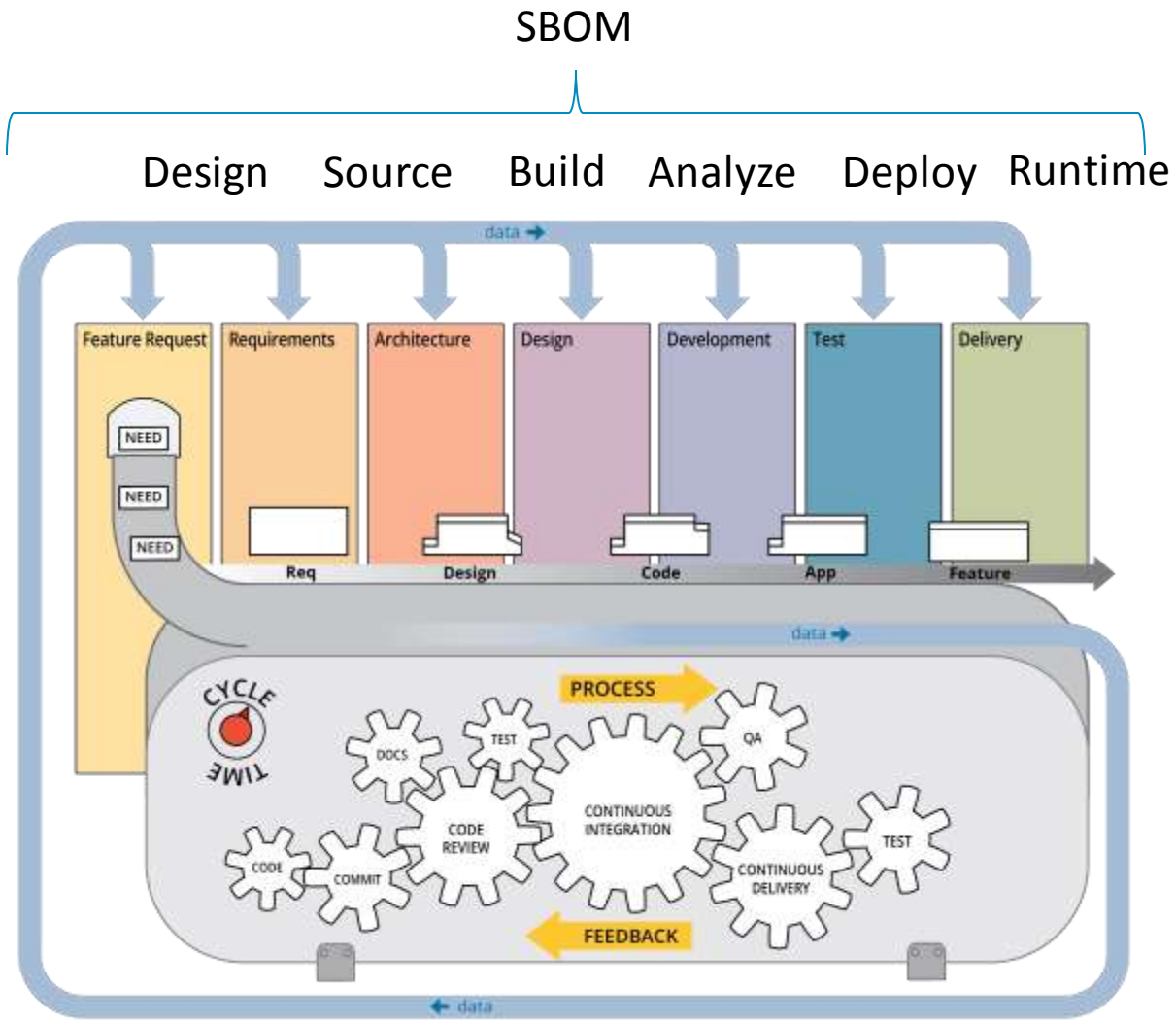


Document the "ingredients" within each software package

- Expediates detection of vulnerable components downstream.
- Avoids heavy costs incurred due to security breaches, a bad reputation, and regulatory penalties.

- Map all Apps to codebases
  - ➢ **Decide:** project team be established to review and update data quality management
- Enforce CI/CD policy compliance
  - ➢ **Decide:** by default, formally require teams follow end-to-end for build and deploy using CI/CD pipeline that supports integration of SBOM generation, *or they file an exception.*

Takeaway!



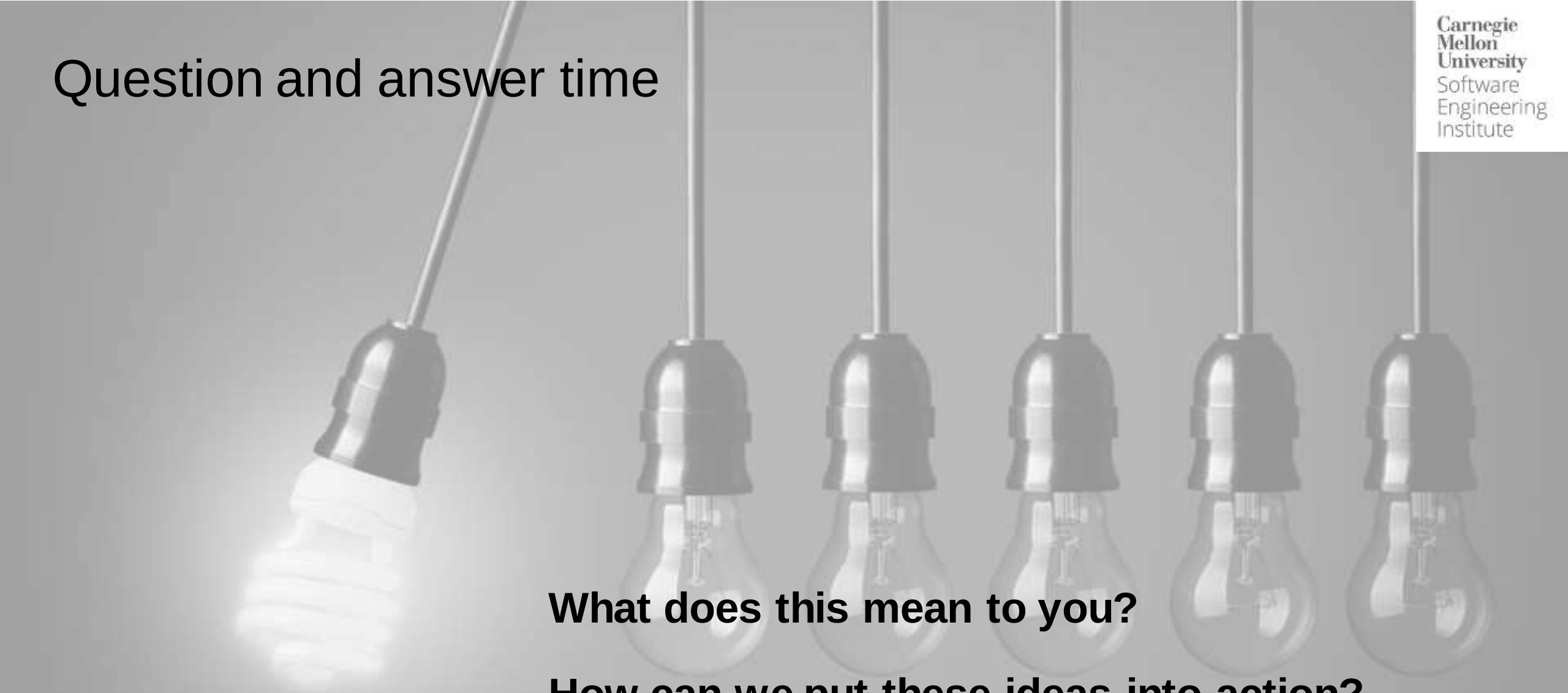
- Develop and share Open source software usage policy
- Bring the SBOM into development early:
- As early as planning stage like creating security stories.
- Evaluate a product's threat resistance
- Create a centralized private repositories of vetted 3rd party components for all developers
- Apply policy and management through DevOps pipeline
- Continuous training and monitoring developers activities
- Automate and monitor dependencies management
- Track build and deploy decencies list
- Apply discovered(new) vulnerabilities and deployment process
- Establish good product distribution practices
- Supplier security commitment evidence
- Automate your SBOM tools as much as possible.
- Integrate. Integrate. **INTEGRATE**!

Carnegie Mellon University Software Engineering Institute

For More Information

DevSecOps: https://www.sei.cmu.edu/go/devops
DevOps Blog: https://insights.sei.cmu.edu/devops
Webinar Series: https://www.sei.cmu.edu/publications/webinars/
Podcast Series: https://www.sei.cmu.edu/publications/podcasts/

Question and answer time



What does this mean to you?

How can we put these ideas into action?

Contact Information

Hasan Yasar
Technical Director, Adjunct Faculty Member
Continuous Deployment of Capability,
Software Engineering Institute | Carnegie Mellon University
hyasar@cmu.edu