(SEC)DEVOPS

Michael Boeynaems

Glenn De Ranter

SPLYNTER

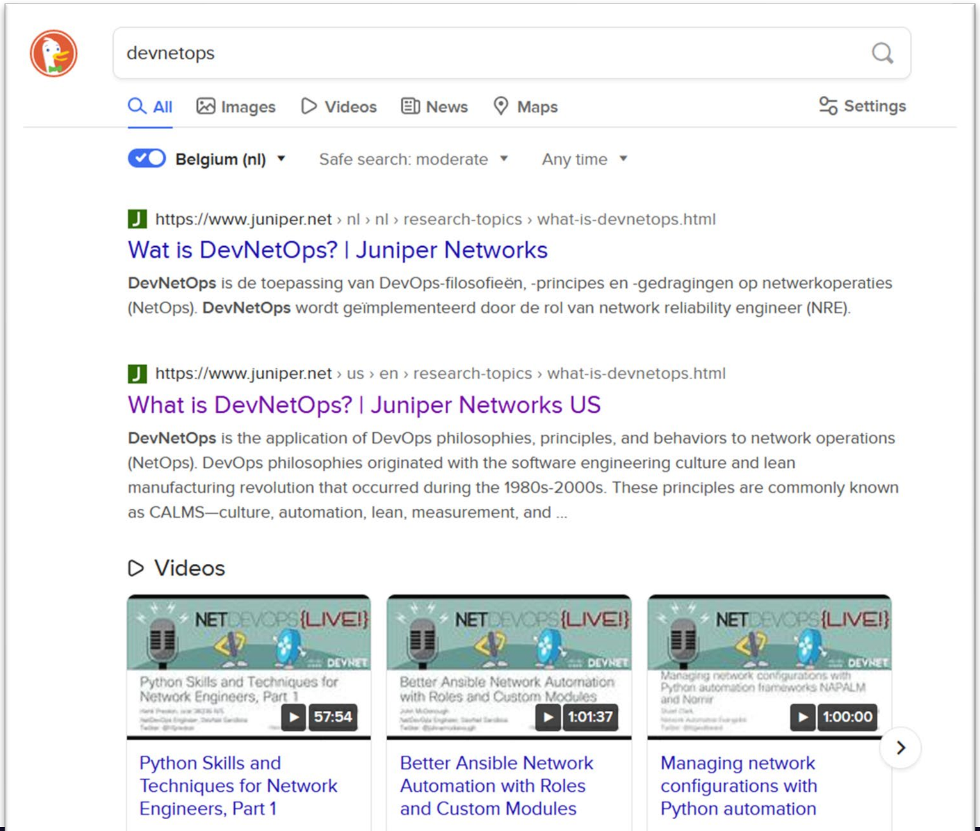
OUTLINE

- What is SecDevops
- Why SecDevOps
- The SecDevOps pipeline
- Increasing maturity

For the security we said we'd hire someone to...
Hold on, that's not the priority. Let's get the project started first, we'll see about that later...
Hey the project is nearly done, we should do a security audit to...
Too late man, we don't have the time or the budget for that
The site's been online for 10 days, and we haven't done any security testing, I'm nervous...
Don't worry, we'll take care of it later...
LATER...
DOWN! THE SITE IS DOWN! WE'VE BEEN HACKED!
How could you let this happen!?
CommitStrip.com

WHAT IS SECDEVOPS

Have you ever heard of ‘DevNetOps’, or ‘DevInfrOps’?



Belgium (nl) ▾ Safe search: moderate ▾ Any time ▾

https://www.juniper.net › nl › nl › research-topics › what-is-devnetops.html

Wat is DevNetOps? | Juniper Networks

DevNetOps is de toepassing van DevOps-filosofieën, -principes en -gedragingen op netwerkoperaties (NetOps). **DevNetOps** wordt geïmplementeerd door de rol van network reliability engineer (NRE).

https://www.juniper.net › us › en › research-topics › what-is-devnetops.html

What is DevNetOps? | Juniper Networks US

DevNetOps is the application of DevOps philosophies, principles, and behaviors to network operations (NetOps). DevOps philosophies originated with the software engineering culture and lean manufacturing revolution that occurred during the 1980s-2000s. These principles are commonly known as CALMS—culture, automation, lean, measurement, and ...

Videos

NETDEVOPS{LIVE!}
DEVNET
Python Skills and Techniques for Network Engineers, Part 1
57:54
NETDEVOPS{LIVE!}
DEVNET
Better Ansible Network Automation with Roles and Custom Modules
1:01:37
NETDEVOPS{LIVE!}
DEVNET
Managing network configurations with Python automation frameworks NAPALM and Nornir
1:00:00

Python Skills and Techniques for Network Engineers, Part 1

Better Ansible Network Automation with Roles and Custom Modules

Managing network configurations with Python automation

SECDEVOPS

"DevOps is a culture, fostering ***collaboration amongst all participants involved*** *in the development and maintenance of software."* [1]

So why do we need a word like 'DevSecOps' or 'SecDevOps'? Security is part of '*all participants*', right?

Anyway…

RuggedOps (DevSecOps): "Rugged" describes software development organisations that have a culture of rapidly evolving their ability to create available, survivable, defensible, secure, and resilient software. [2]

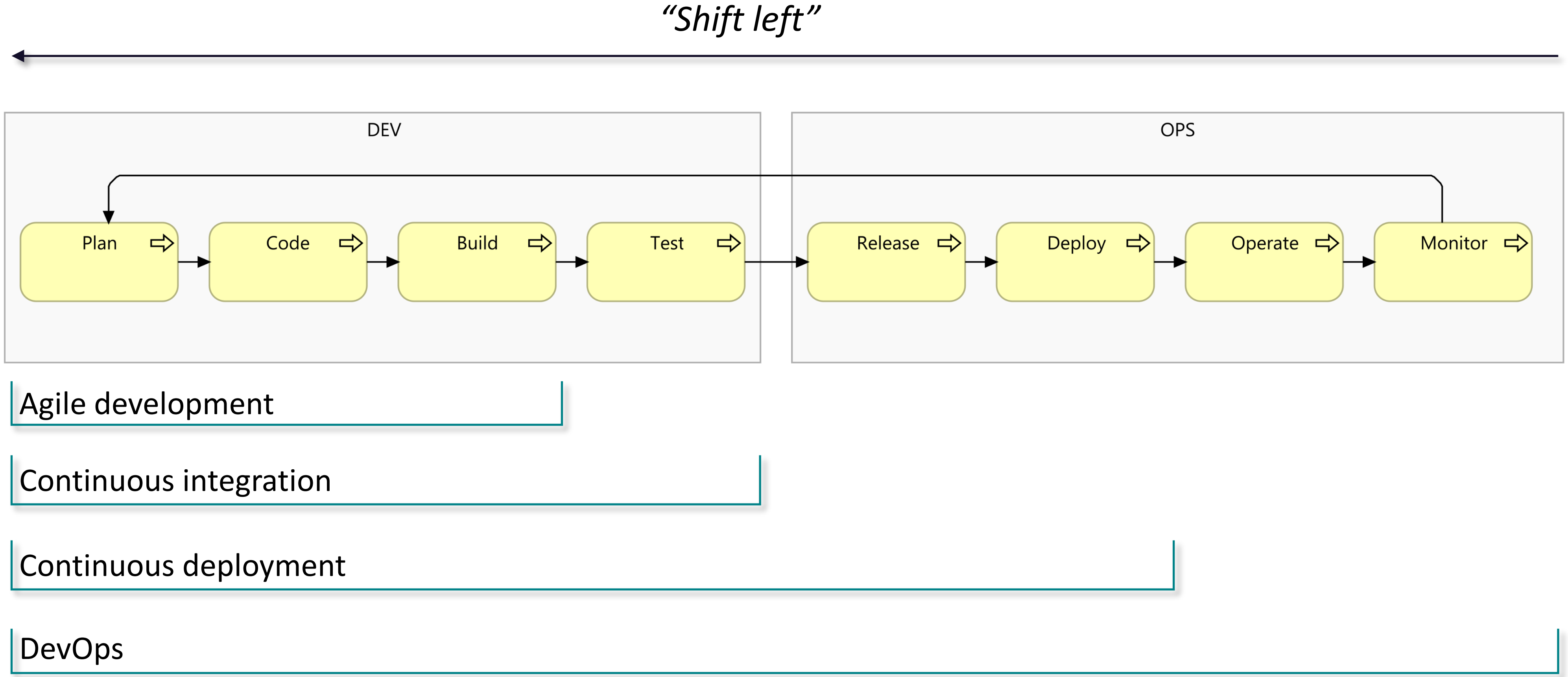
WHAT IS DEVOPS?

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.



"Shift left"

DEV

OPS

Plan → Code → Build → Test → Release → Deploy → Operate → Monitor

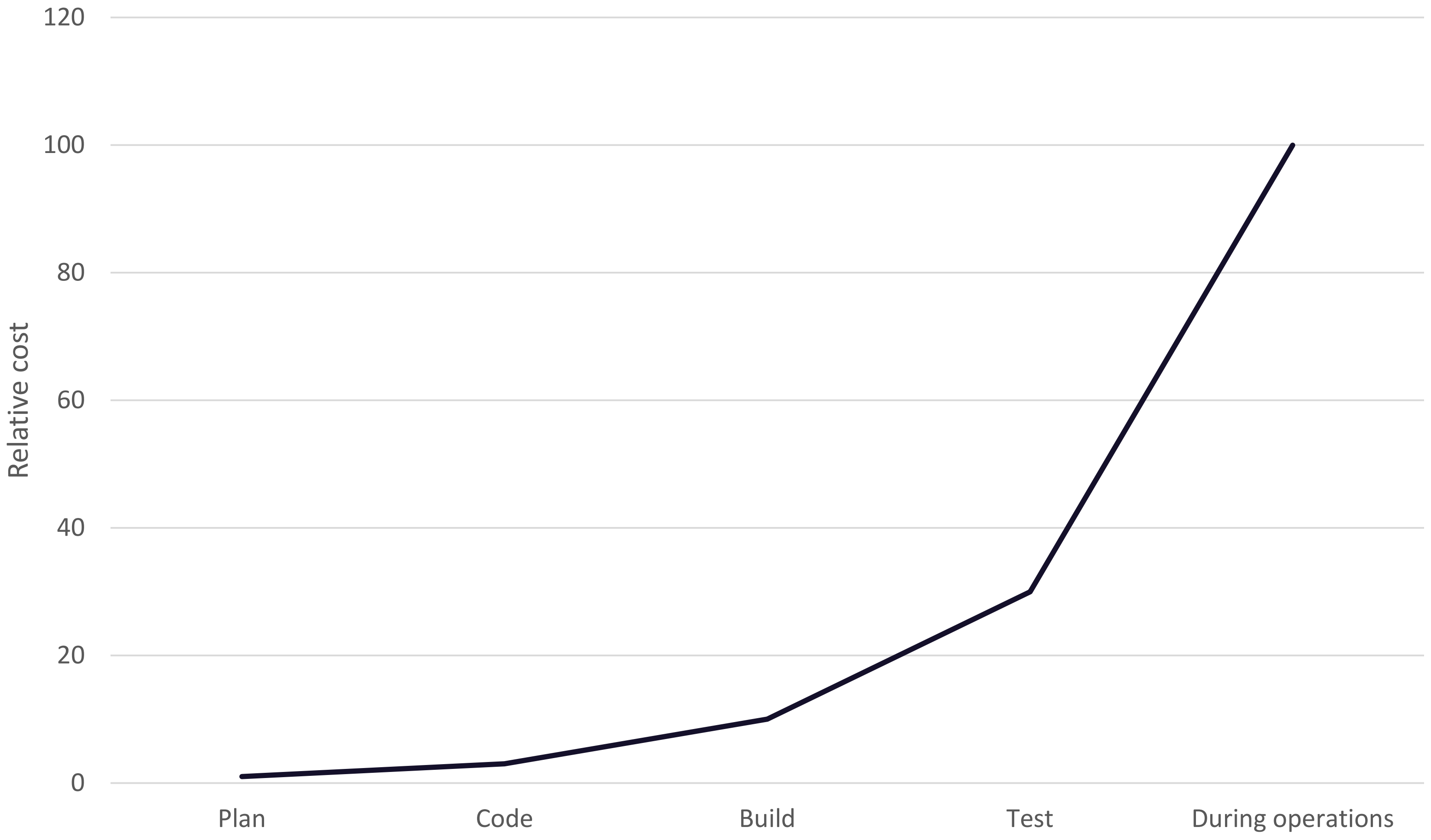
Agile development

Continuous integration

Continuous deployment

DevOps

Cost of fixing a software defect



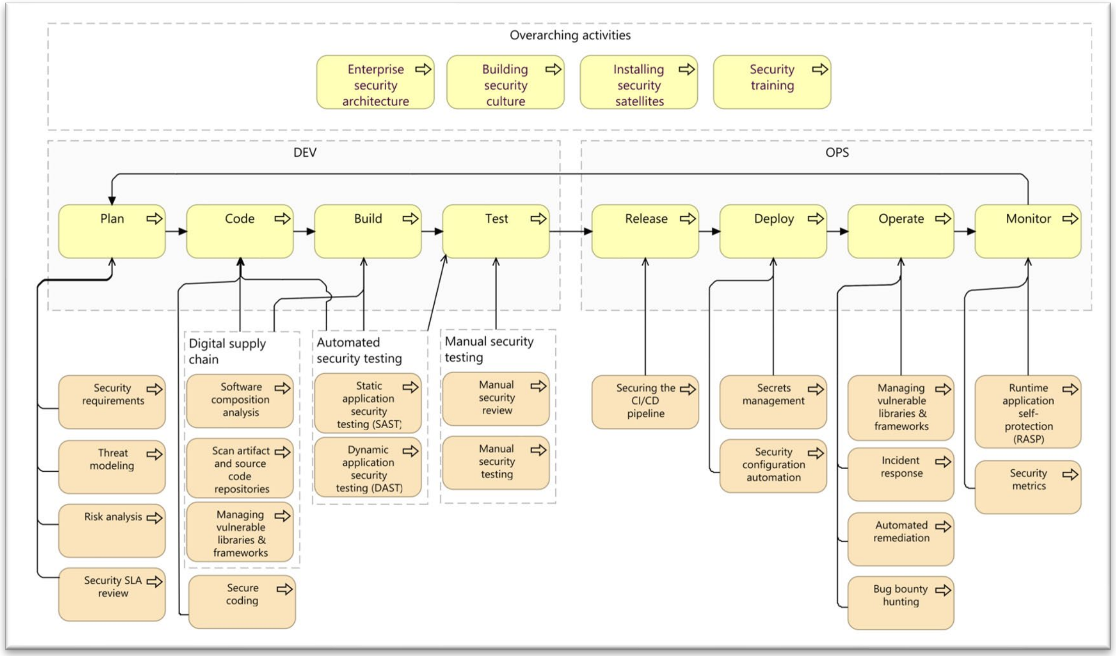
WHY SECDEVOPS

SECDEVOPS – WHY?

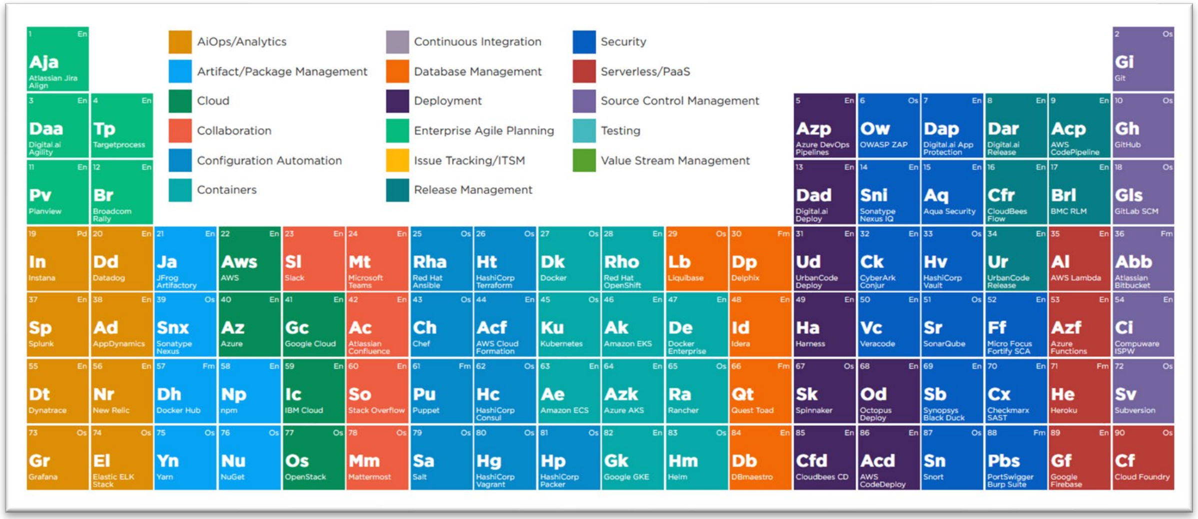
- Greater speed and agility for security teams
- An ability to rapidly respond to changes and needs
- Better collaboration and communication among teams
- More opportunities for automated builds as well as quality and security testing
- Early identification of vulnerabilities in code
- Automation to free up team member assets to work on high-value tasks

THE SECDEVOPS PIPELINE

BUILDING A SECDEVOPS PIPELINE

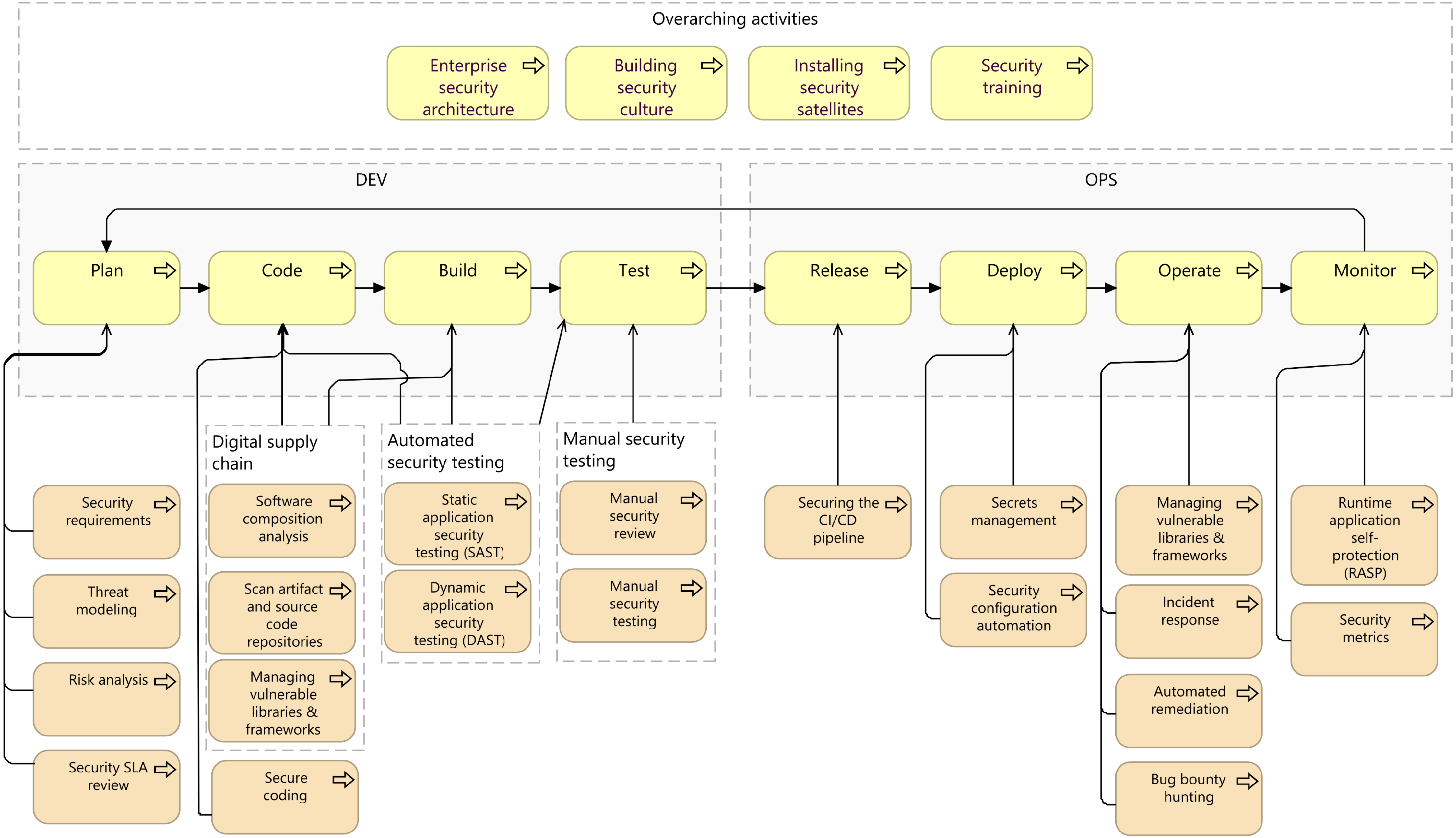

Overarching activities
Enterprise security architecture
Building security culture
Installing security satellites
Security training
DEV
OPS
Plan
Code
Build
Test
Release
Deploy
Operate
Monitor
Digital supply chain
Automated security testing
Manual security testing
Security requirements
Threat modeling
Risk analysis
Security SLA review
Software composition analysis
Scan artifact and source code repositories
Managing vulnerable libraries & frameworks
Secure coding
Static application security testing (SAST)
Dynamic application security testing (DAST)
Manual security review
Manual security testing
Securing the CI/CD pipeline
Secrets management
Security configuration automation
Managing vulnerable libraries & frameworks
Incident response
Automated remediation
Bug bounty hunting
Runtime application self-protection (RASP)
Security metrics

>



[3]

Focus on capabilities, not on tools (there are **many** tools out there)



Overarching activities

Enterprise security architecture

Building security culture

Installing security satellites

Security training

DEV

OPS

Plan

Code

Build

Test

Release

Deploy

Operate

Monitor

Security requirements

Threat modeling

Risk analysis

Security SLA review

Digital supply chain

Software composition analysis

Scan artifact and source code repositories

Managing vulnerable libraries & frameworks

Secure coding

Automated security testing

Static application security testing (SAST)

Dynamic application security testing (DAST)

Manual security testing

Manual security review

Manual security testing

Securing the CI/CD pipeline

Secrets management

Security configuration automation

Managing vulnerable libraries & frameworks

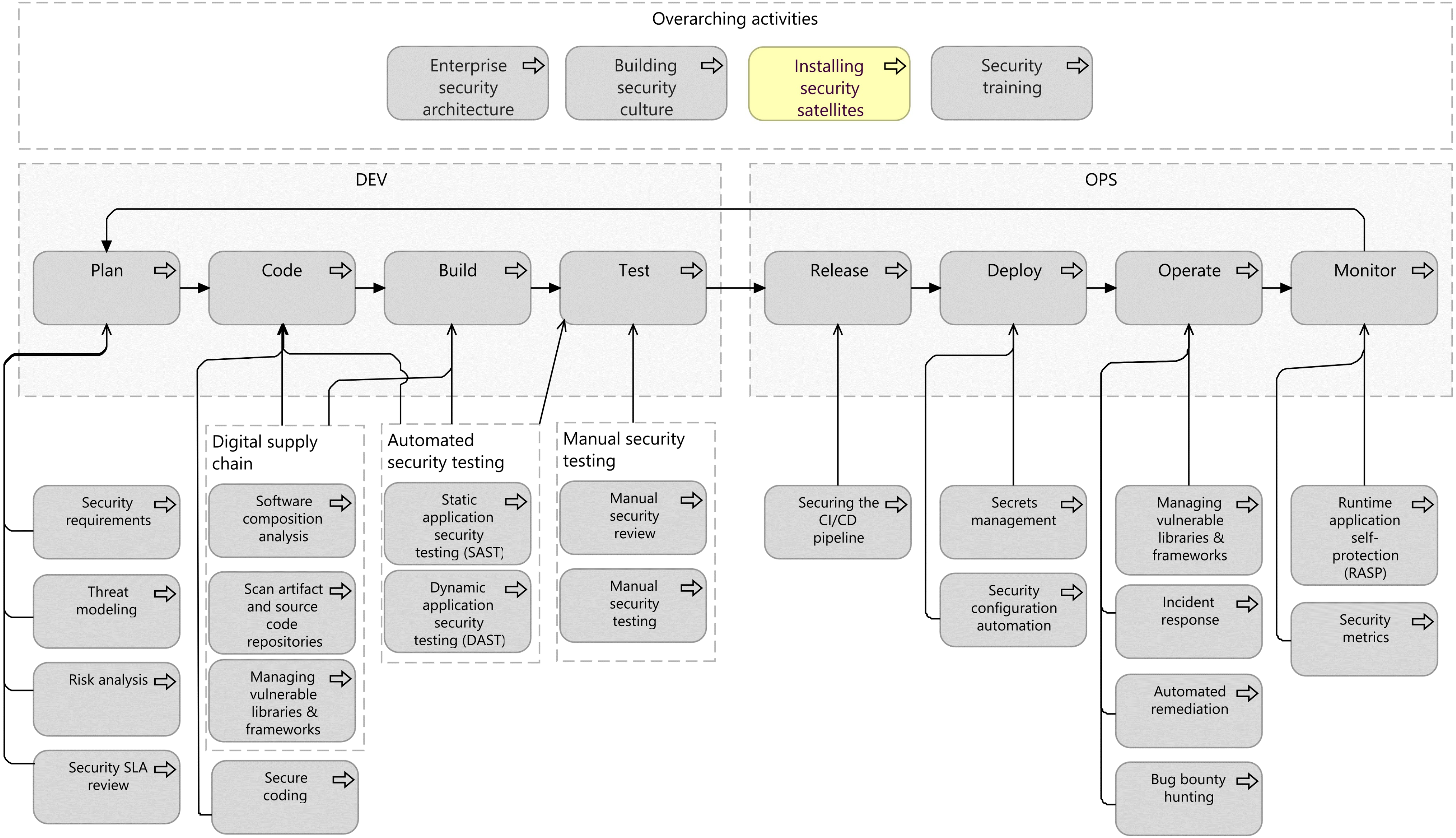
Incident response

Automated remediation

Bug bounty hunting

Runtime application self-protection (RASP)

Security metrics



Overarching activities

Enterprise security architecture

Building security culture

Installing security satellites

Security training

DEV

OPS

Plan

Code

Build

Test

Release

Deploy

Operate

Monitor

Security requirements

Threat modeling

Risk analysis

Security SLA review

Digital supply chain

Software composition analysis

Scan artifact and source code repositories

Managing vulnerable libraries & frameworks

Secure coding

Automated security testing

Static application security testing (SAST)

Dynamic application security testing (DAST)

Manual security testing

Manual security review

Manual security testing

Securing the CI/CD pipeline

Secrets management

Security configuration automation

Managing vulnerable libraries & frameworks

Incident response

Automated remediation

Bug bounty hunting

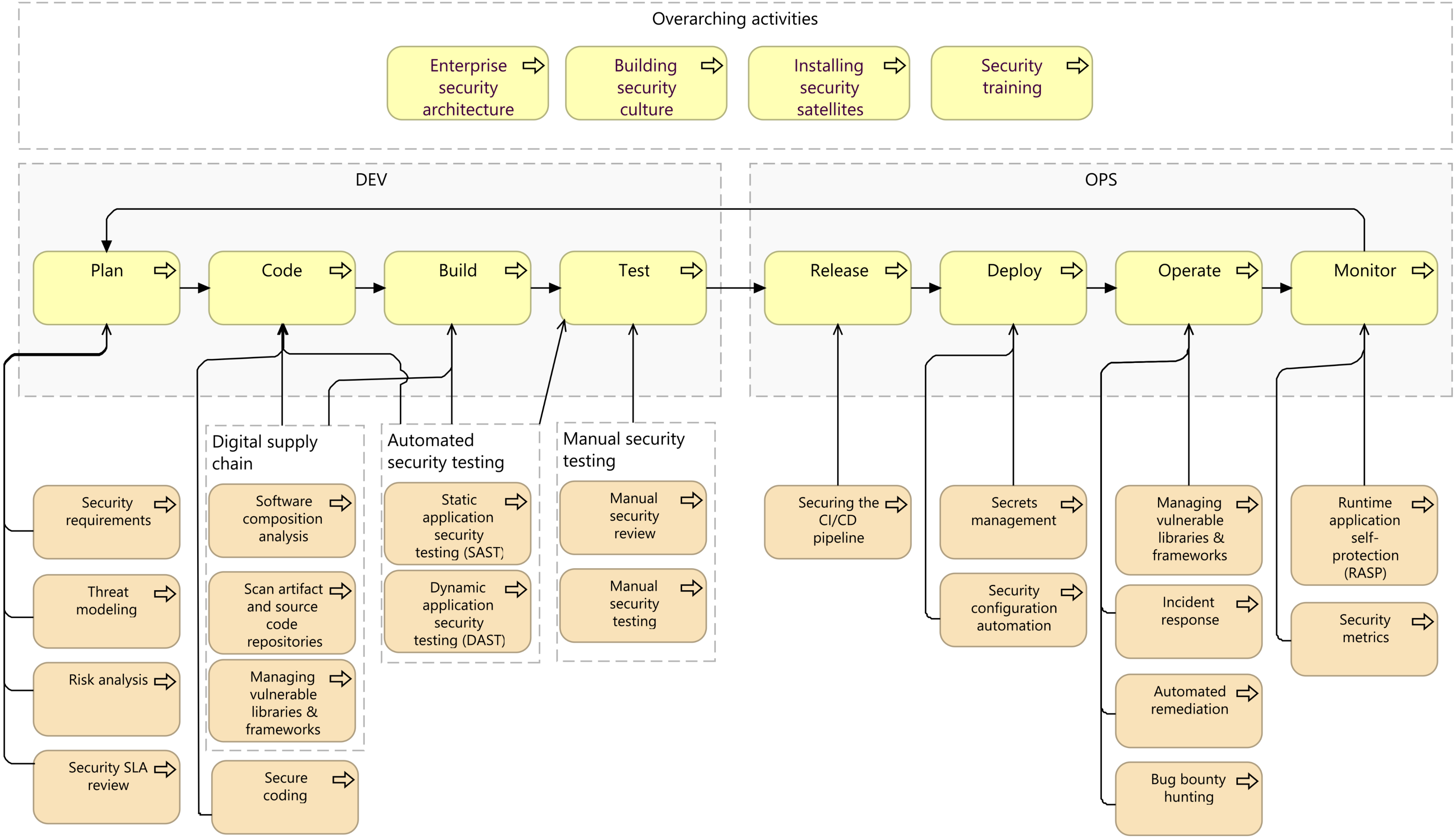
Runtime application self-protection (RASP)

Security metrics

SECURITY SATELLITES - EXPECTATIONS

- Sharing knowledge
- Helping in decision-making for new features
- Building threat models
- Monitoring best practices

Can you spot the security champion in this team?



Overarching activities

Enterprise security architecture

Building security culture

Installing security satellites

Security training

DEV

OPS

Plan

Code

Build

Test

Release

Deploy

Operate

Monitor

Security requirements

Threat modeling

Risk analysis

Security SLA review

Digital supply chain

Software composition analysis

Scan artifact and source code repositories

Managing vulnerable libraries & frameworks

Secure coding

Automated security testing

Static application security testing (SAST)

Dynamic application security testing (DAST)

Manual security testing

Manual security review

Manual security testing

Securing the CI/CD pipeline

Secrets management

Security configuration automation

Managing vulnerable libraries & frameworks

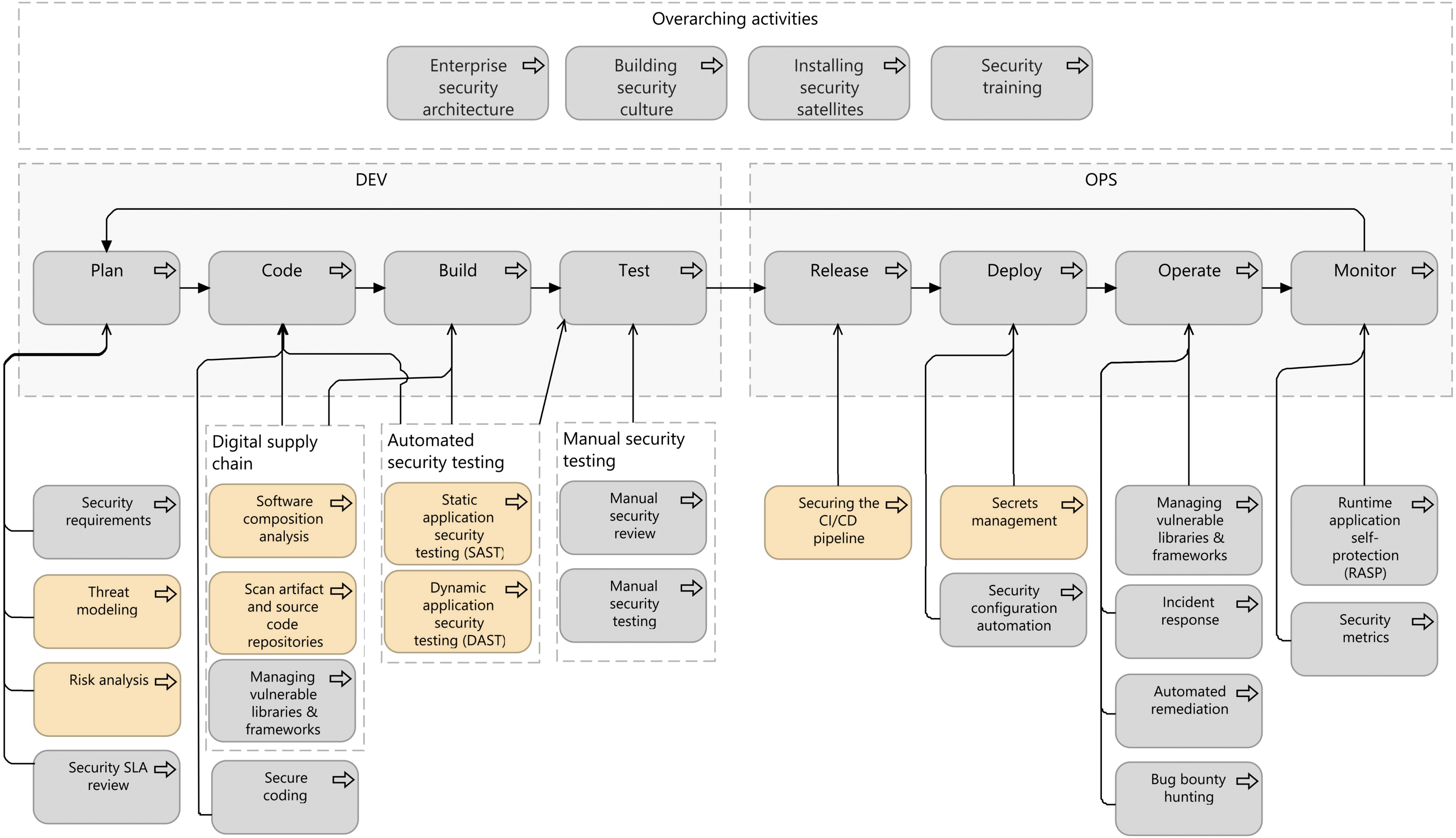
Incident response

Automated remediation

Bug bounty hunting

Runtime application self-protection (RASP)

Security metrics



Overarching activities

Enterprise security architecture

Building security culture

Installing security satellites

Security training

DEV

OPS

Plan

Code

Build

Test

Release

Deploy

Operate

Monitor

Security requirements

Threat modeling

Risk analysis

Security SLA review

Digital supply chain

Software composition analysis

Scan artifact and source code repositories

Managing vulnerable libraries & frameworks

Secure coding

Automated security testing

Static application security testing (SAST)

Dynamic application security testing (DAST)

Manual security testing

Manual security review

Manual security testing

Securing the CI/CD pipeline

Secrets management

Security configuration automation

Managing vulnerable libraries & frameworks

Incident response

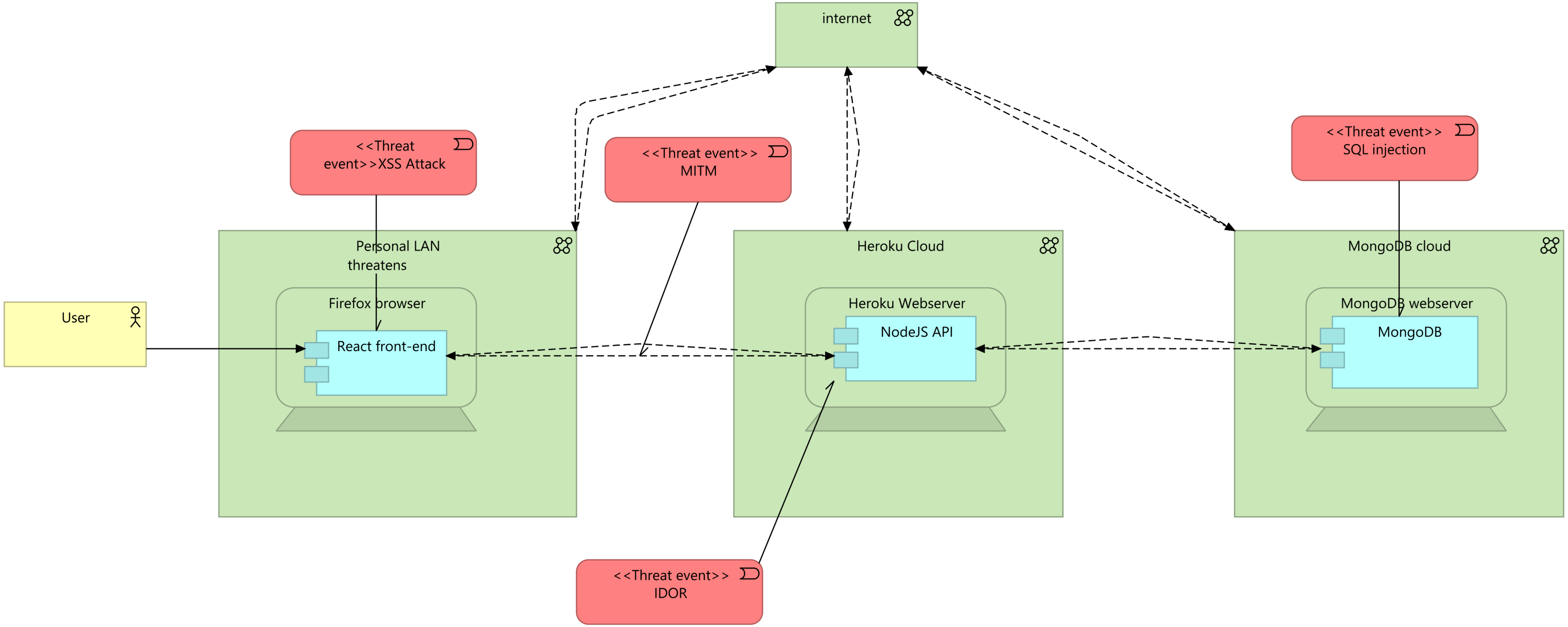
Automated remediation

Bug bounty hunting

Runtime application self-protection (RASP)

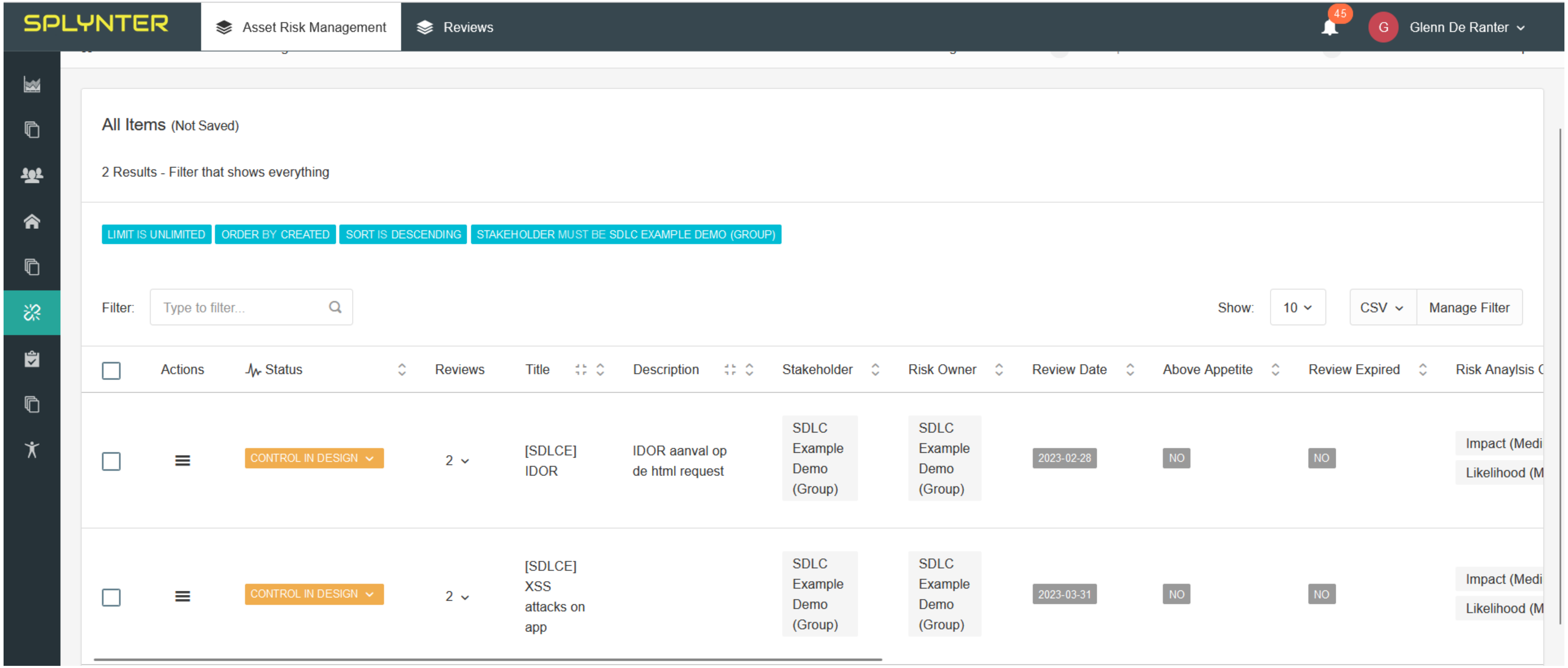
Security metrics

Threat modeling



Demo Threat model

Risk assessment



All Items (Not Saved)

2 Results - Filter that shows everything

LIMIT IS UNLIMITED | ORDER BY CREATED | SORT IS DESCENDING | STAKEHOLDER MUST BE SDLC EXAMPLE DEMO (GROUP)

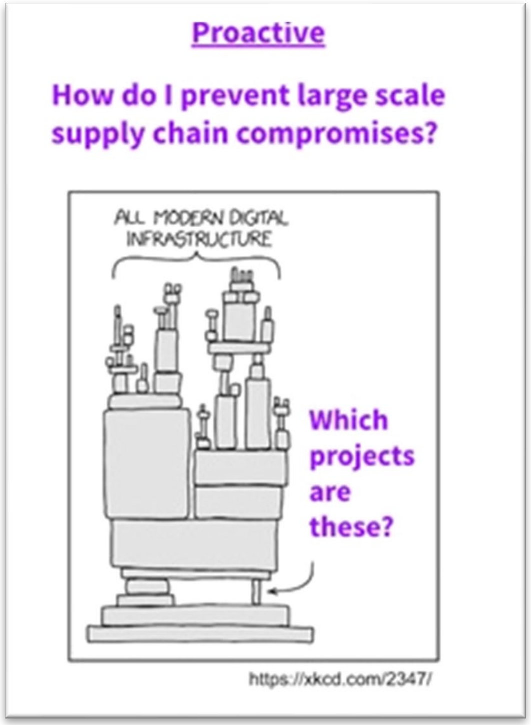
Filter: Type to filter...

Show: 10

CSV | Manage Filter

□	Actions	Status	Reviews	Title	Description	Stakeholder	Risk Owner	Review Date	Above Appetite	Review Expired	Risk Anaylsis (
□	≡	CONTROL IN DESIGN	2	[SDLCE] IDOR	IDOR aanval op de html request	SDLC Example Demo (Group)	SDLC Example Demo (Group)	2023-02-28	NO	NO	Impact (Medi Likelihood (M
□	≡	CONTROL IN DESIGN	2	[SDLCE] XSS attacks on app		SDLC Example Demo (Group)	SDLC Example Demo (Group)	2023-03-31	NO	NO	Impact (Medi Likelihood (M

Software Composition Analysis (SCA)



Proactive

How do I prevent large scale supply chain compromises?

Which projects are these?

https://xkcd.com/2347/

SLSA

trivy

syft

Static Application Security Testing (SAST)

*“SAST is a frequently used Application Security (AppSec) tool, **which scans an application’s source, binary, or byte code**. A white-box testing tool, it identifies the root cause of vulnerabilities and helps remediate the underlying security flaws.”*[4]

SpotBugs

Dynamic Application Security Testing (DAST)

*“DAST is the process of analyzing a web application through the front-end to **find vulnerabilities through simulated attacks**.”*[4]

OWASP
Zed Attack Proxy

BURPSUITE

Securing the CI/CD pipeline

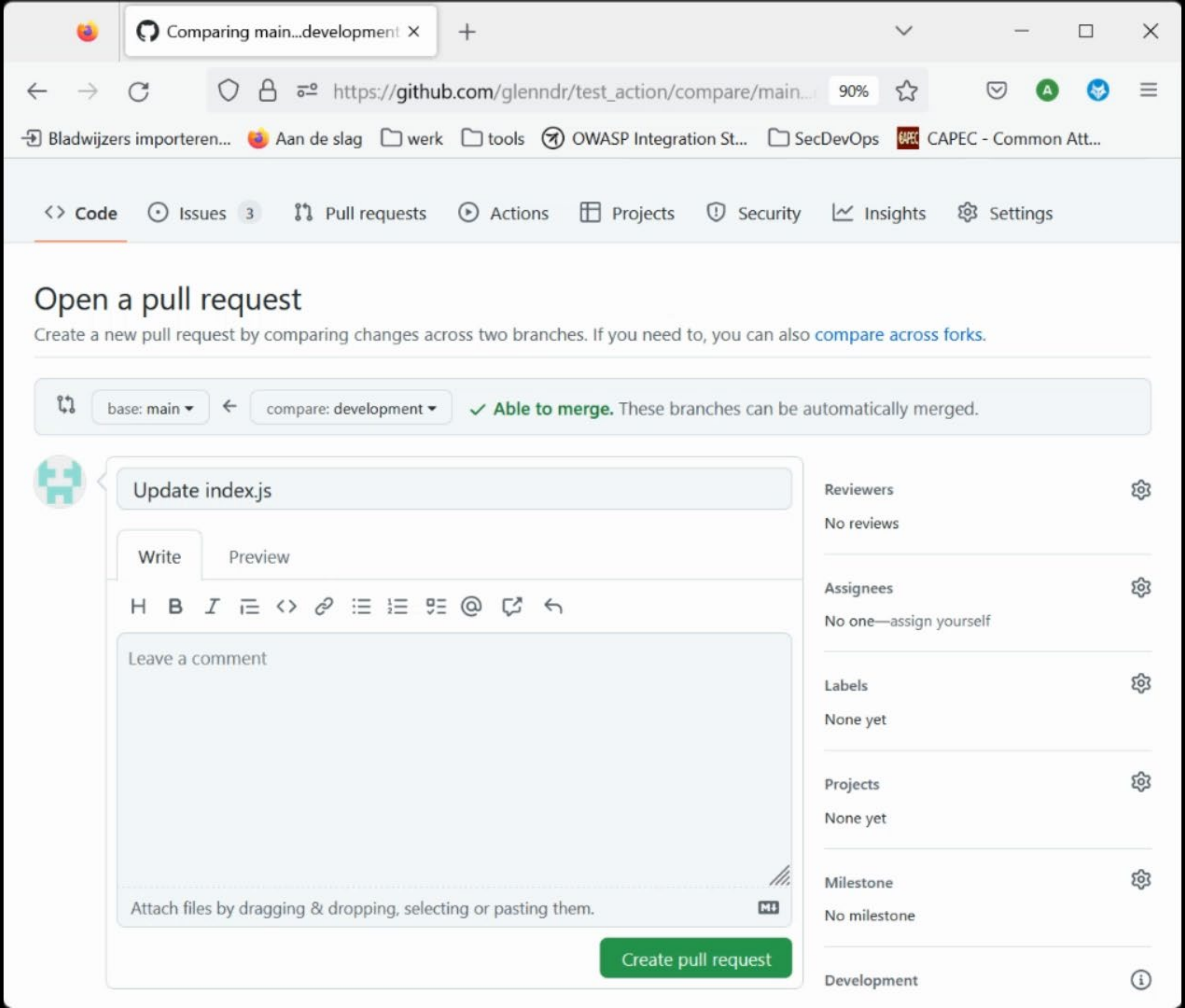
"Infrastructure as Code (IaC) automates the provisioning of infrastructure, enabling your organization to develop, deploy, and scale cloud applications with greater speed, less risk, and reduced cost." [5]

GitHub Jenkins Azure DevOps

Secret management

"Secret management allows to safely and securely store secret values and (binary) files such as passwords, tokens, Key files etc.,) in central repository or system. Through Access Control Lists ONLY specific entities can be retrieved or decrypted."[6]

HashiCorp Vault



Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

base: main ← compare: development ✓ **Able to merge.** These branches can be automatically merged.

Update index.js

Write
Preview
H B I
Leave a comment
Attach files by dragging & dropping, selecting or pasting them.
Create pull request

Reviewers
No reviews

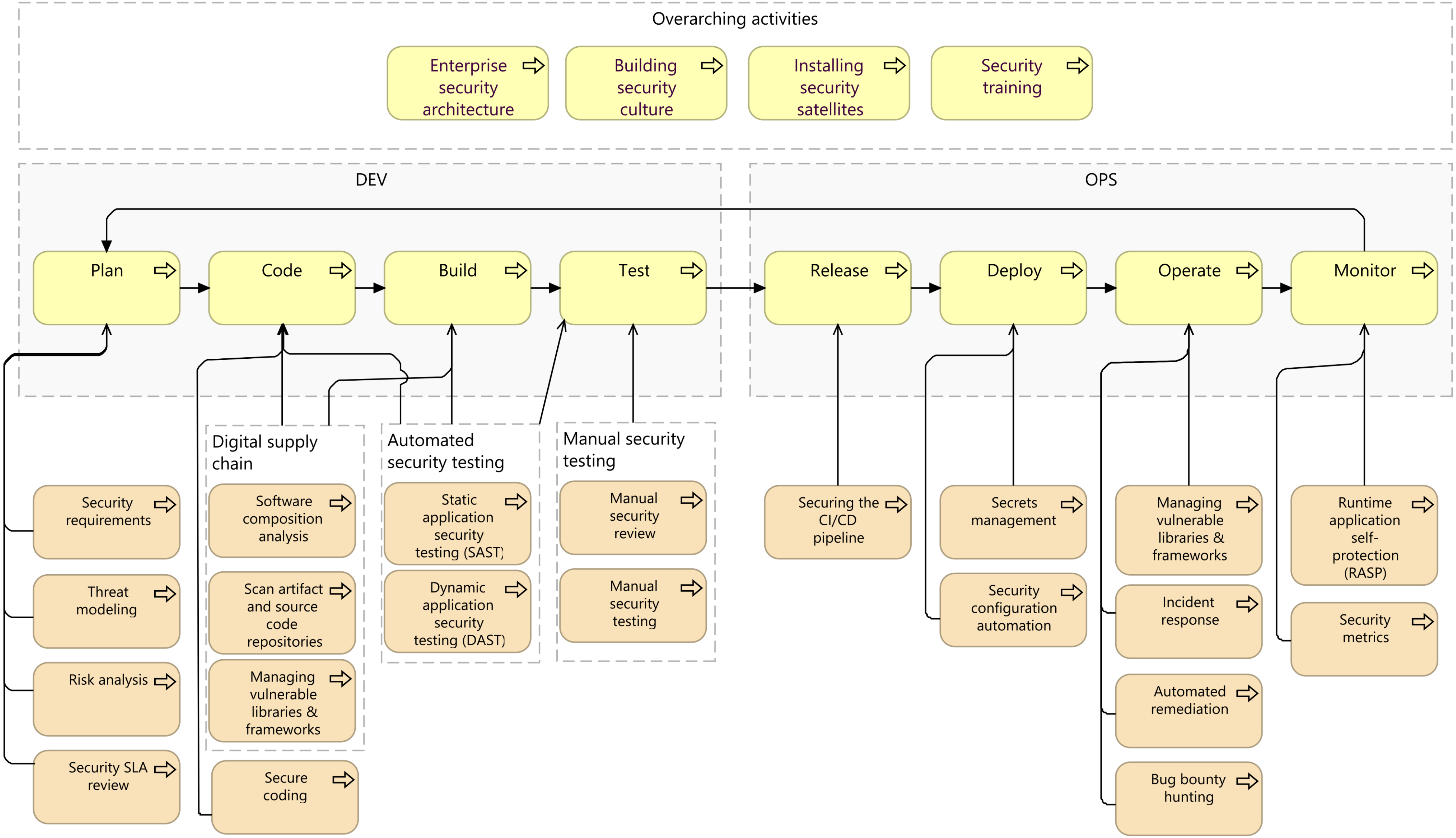
Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

Development



Overarching activities

Enterprise security architecture

Building security culture

Installing security satellites

Security training

DEV

OPS

Plan

Code

Build

Test

Release

Deploy

Operate

Monitor

Security requirements

Threat modeling

Risk analysis

Security SLA review

Digital supply chain

Software composition analysis

Scan artifact and source code repositories

Managing vulnerable libraries & frameworks

Secure coding

Automated security testing

Static application security testing (SAST)

Dynamic application security testing (DAST)

Manual security testing

Manual security review

Manual security testing

Securing the CI/CD pipeline

Secrets management

Security configuration automation

Managing vulnerable libraries & frameworks

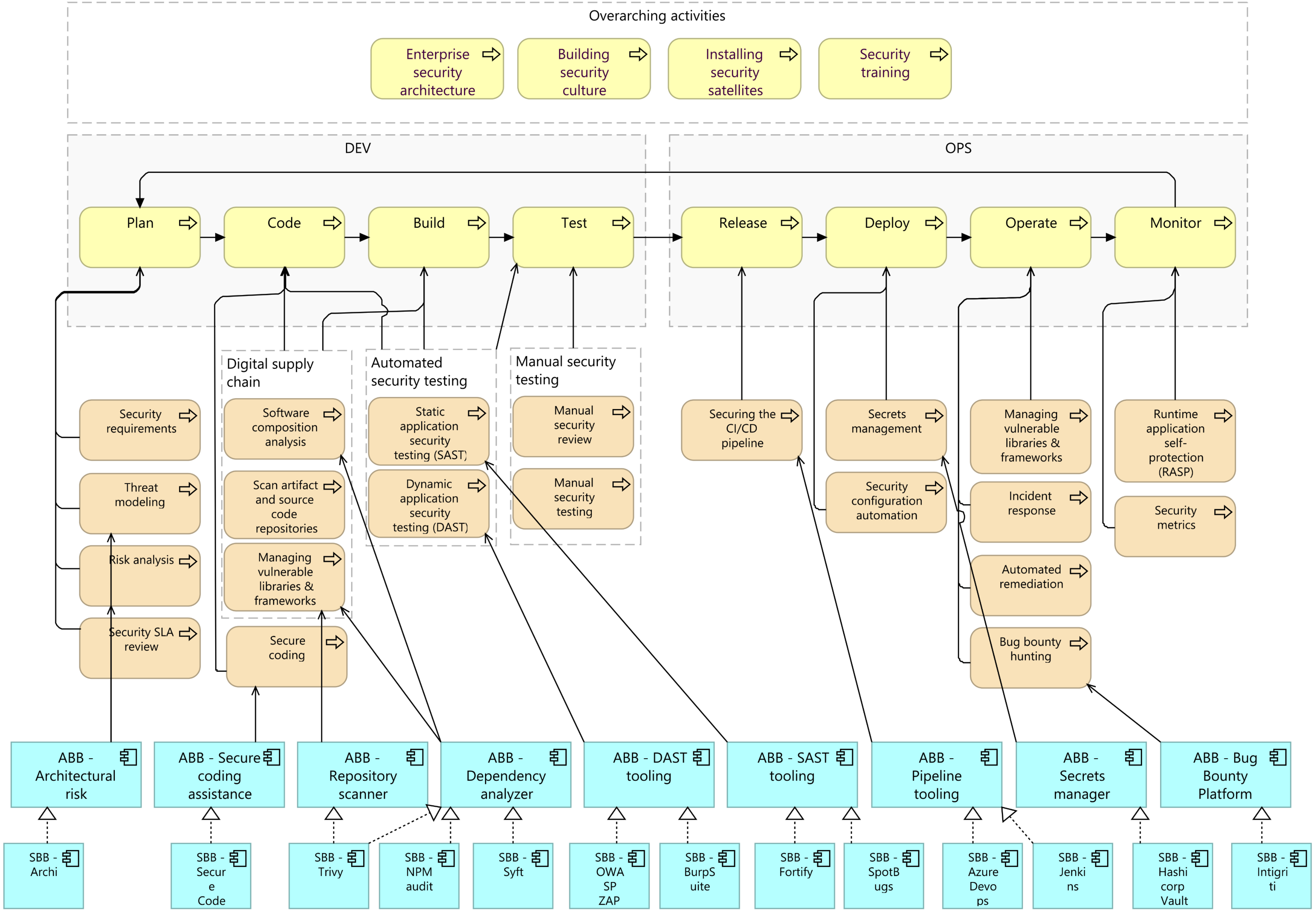
Incident response

Automated remediation

Bug bounty hunting

Runtime application self-protection (RASP)

Security metrics



Overarching activities

Enterprise security architecture

Building security culture

Installing security satellites

Security training

DEV

OPS

Plan

Code

Build

Test

Release

Deploy

Operate

Monitor

Security requirements

Threat modeling

Risk analysis

Security SLA review

Digital supply chain

Software composition analysis

Scan artifact and source code repositories

Managing vulnerable libraries & frameworks

Secure coding

Automated security testing

Static application security testing (SAST)

Dynamic application security testing (DAST)

Manual security testing

Manual security review

Manual security testing

Securing the CI/CD pipeline

Secrets management

Security configuration automation

Managing vulnerable libraries & frameworks

Incident response

Automated remediation

Bug bounty hunting

Runtime application self-protection (RASP)

Security metrics

ABB - Architectural risk

ABB - Secure coding assistance

ABB - Repository scanner

ABB - Dependency analyzer

ABB - DAST tooling

ABB - SAST tooling

ABB - Pipeline tooling

ABB - Secrets manager

ABB - Bug Bounty Platform

SBB - Archi

SBB - Secure Code

SBB - Trivy

SBB - NPM audit

SBB - Syft

SBB - OWASP ZAP

SBB - BurpSuite

SBB - Fortify

SBB - SpotBugs

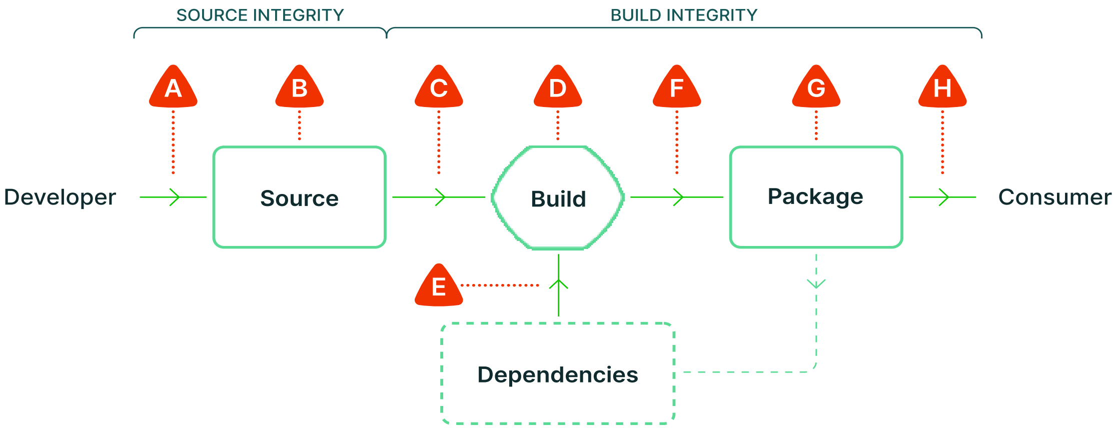
SBB - Azure Devops

SBB - Jenkins

SBB - Hashicorp Vault

SBB - Intigriti

How do you ensure that the code being deployed to production in binary form is the same one that went through your pipeline?

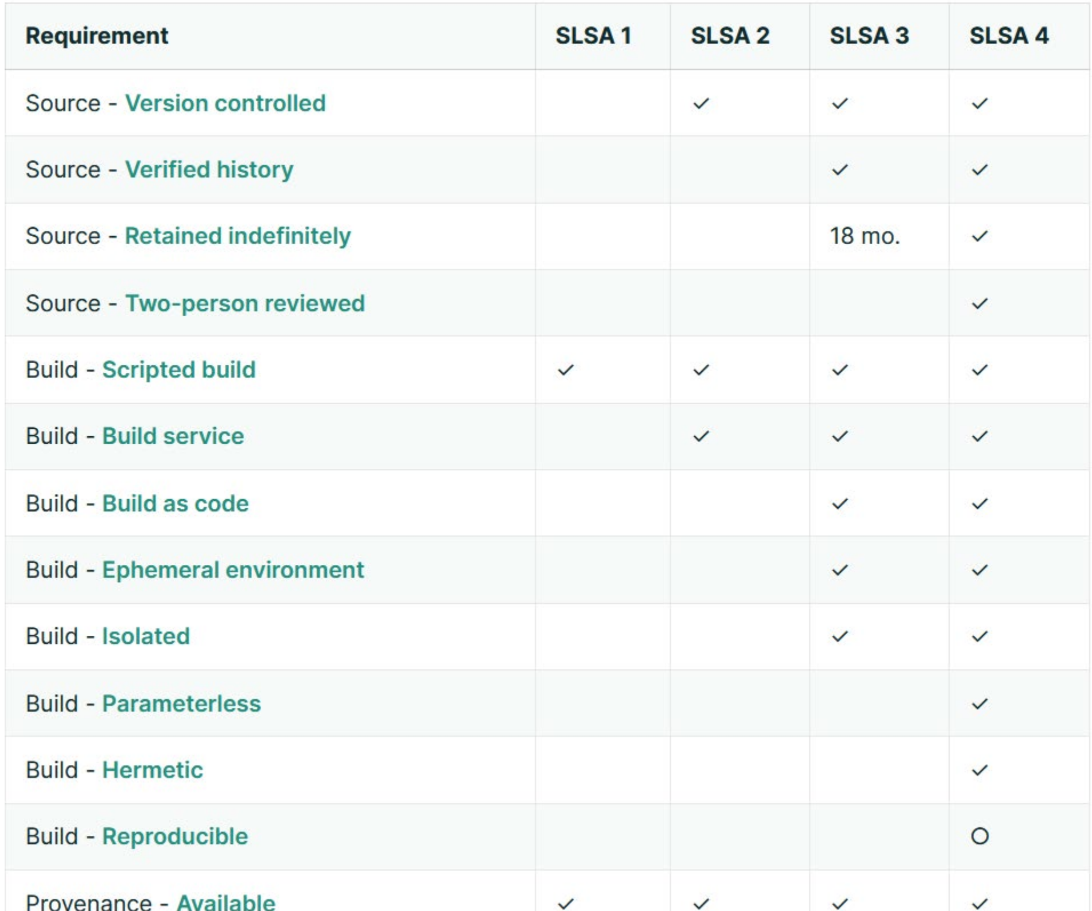


A Submit unauthorized change
B Compromise source repo

C Build from modified source
D Compromise build process
E Use compromised dependency

F Upload modified package
G Compromise package repo
H Use compromised package

[7]

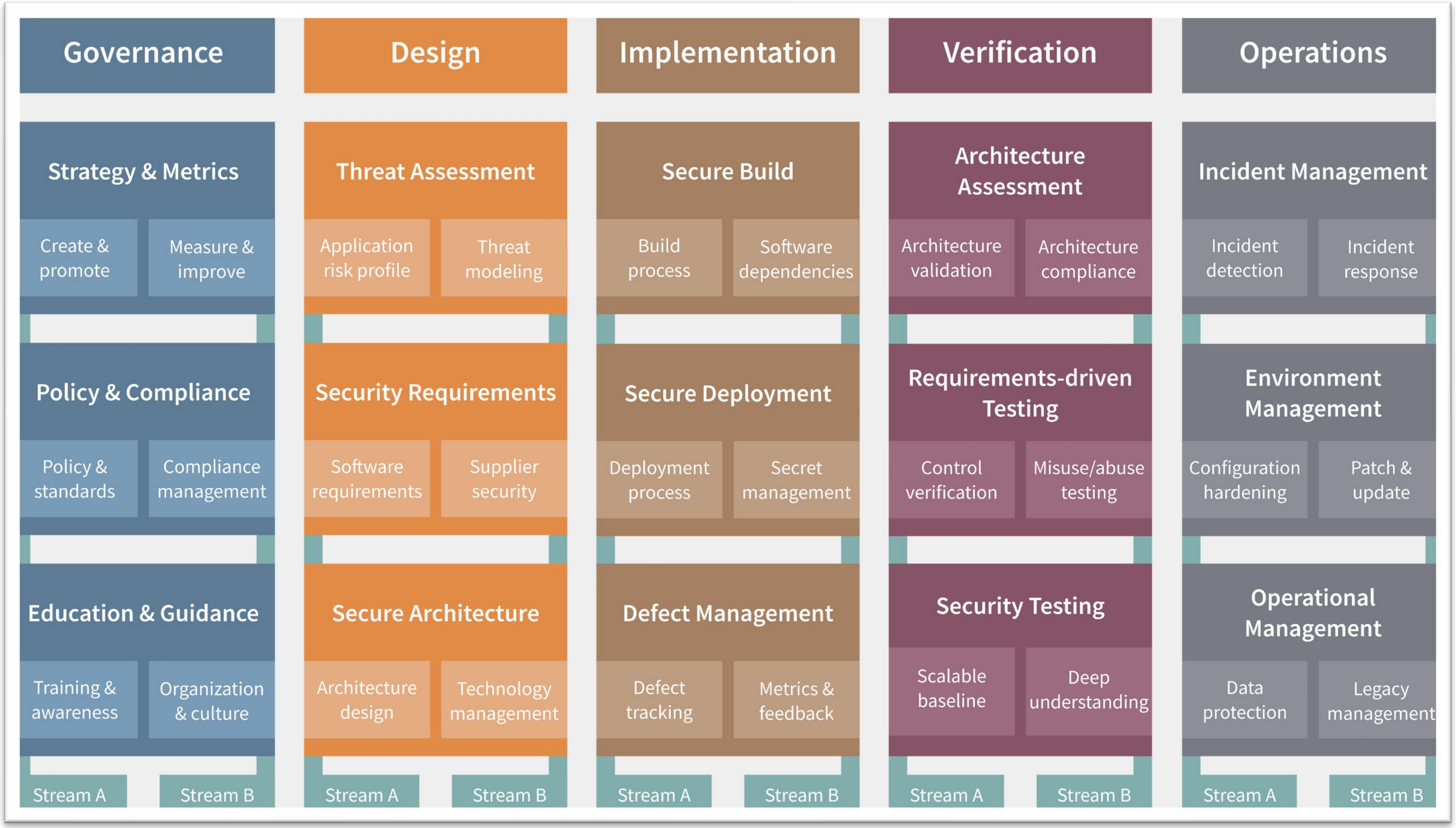


Requirement	SLSA 1	SLSA 2	SLSA 3	SLSA 4
Source - Version controlled		✓	✓	✓
Source - Verified history			✓	✓
Source - Retained indefinitely			18 mo.	✓
Source - Two-person reviewed				✓
Build - Scripted build	✓	✓	✓	✓
Build - Build service		✓	✓	✓
Build - Build as code			✓	✓
Build - Ephemeral environment			✓	✓
Build - Isolated			✓	✓
Build - Parameterless				✓
Build - Hermetic				✓
Build - Reproducible				○
Provenance - Available	✓	✓	✓	✓

[7]

INCREASING MATURITY

SOFTWARE ASSURANCE MATURITY MODEL (SAMM)



Governance		Design		Implementation		Verification		Operations	
Strategy & Metrics		Threat Assessment		Secure Build		Architecture Assessment		Incident Management	
Create & promote	Measure & improve	Application risk profile	Threat modeling	Build process	Software dependencies	Architecture validation	Architecture compliance	Incident detection	Incident response
Policy & Compliance		Security Requirements		Secure Deployment		Requirements-driven Testing		Environment Management	
Policy & standards	Compliance management	Software requirements	Supplier security	Deployment process	Secret management	Control verification	Misuse/abuse testing	Configuration hardening	Patch & update
Education & Guidance		Secure Architecture		Defect Management		Security Testing		Operational Management	
Training & awareness	Organization & culture	Architecture design	Technology management	Defect tracking	Metrics & feedback	Scalable baseline	Deep understanding	Data protection	Legacy management
Stream A	Stream B	Stream A	Stream B	Stream A	Stream B	Stream A	Stream B	Stream A	Stream B

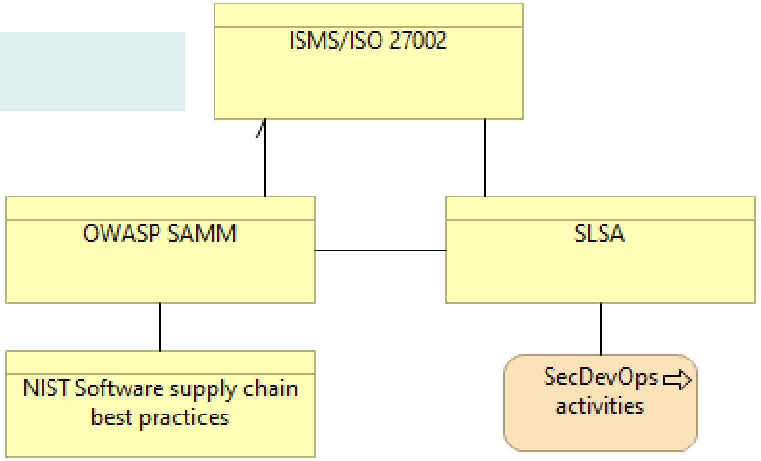
[8]

LINKING EVERYTHING TOGETHER – FUTURE WORK?

Policy frameworks

Maturity models

Best practices and activities



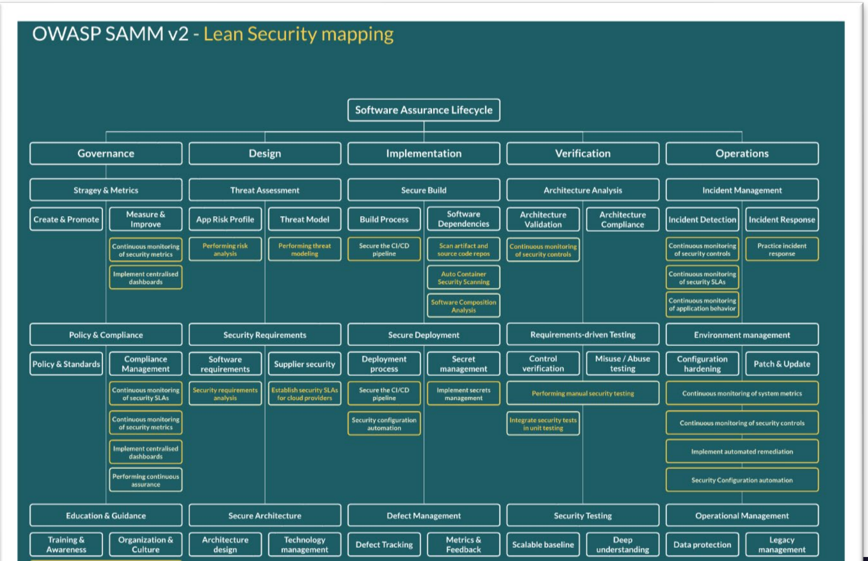
ISMS/ISO 27002

OWASP SAMM

SLSA

NIST Software supply chain best practices

SecDevOps activities

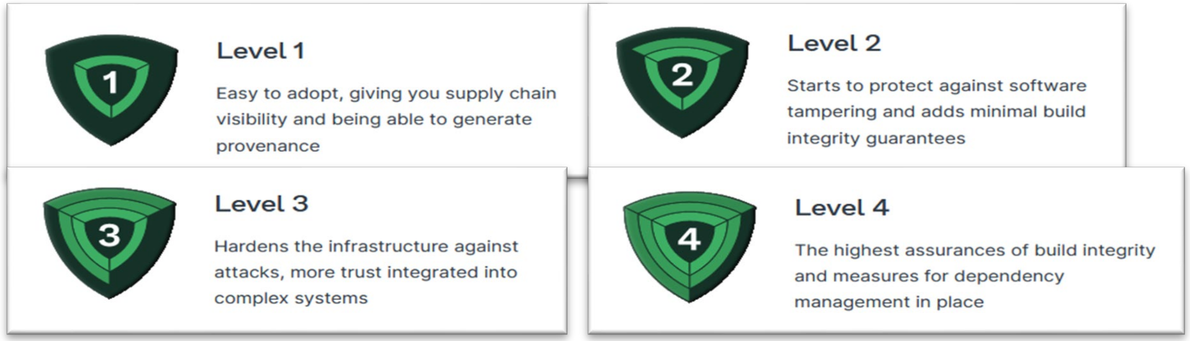


OWASP SAMM v2 - Lean Security mapping

Software Assurance Lifecycle

- Governance
  - Stragey & Metrics
    - Create & Promote
    - Measure & Improve
      - Continuous monitoring of security metrics
      - Implement centralised dashboards
  - Policy & Compliance
    - Policy & Standards
    - Compliance Management
      - Continuous monitoring of security SLAs
      - Continuous monitoring of security metrics
      - Implement centralised dashboards
      - Performing continuous assurance
  - Education & Guidance
    - Training & Awareness
    - Organization & Culture
- Design
  - Threat Assessment
    - App Risk Profile
      - Performing risk analysis
    - Threat Model
      - Performing threat modeling
  - Security Requirements
    - Software requirements
      - Security requirements analysis
    - Supplier security
      - Establish security SLAs for cloud providers
  - Secure Architecture
    - Architecture design
    - Technology management
- Implementation
  - Secure Build
    - Build Process
      - Secure the CI/CD pipeline
    - Software Dependencies
      - Scan artifact and source code repos
      - Auto Container Security Scanning
      - Software Composition Analysis
  - Secure Deployment
    - Deployment process
      - Secure the CI/CD pipeline
      - Security configuration automation
    - Secret management
      - Implement secrets management
  - Defect Management
    - Defect Tracking
    - Metrics & Feedback
- Verification
  - Architecture Analysis
    - Architecture Validation
      - Continuous monitoring of security controls
    - Architecture Compliance
  - Requirements-driven Testing
    - Control verification
      - Integrate security tests in unit testing
    - Misuse / Abuse testing
    - Performing manual security testing
  - Security Testing
    - Scalable baseline
    - Deep understanding
- Operations
  - Incident Management
    - Incident Detection
      - Continuous monitoring of security controls
      - Continuous monitoring of security SLAs
      - Continuous monitoring of application behavior
    - Incident Response
      - Practice incident response
  - Environment management
    - Configuration hardening
    - Patch & Update
    - Continuous monitoring of system metrics
    - Continuous monitoring of security controls
    - Implement automated remediation
    - Security Configuration automation
  - Operational Management
    - Data protection
    - Legacy management

[2]



Level 1

Easy to adopt, giving you supply chain visibility and being able to generate provenance

Level 2

Starts to protect against software tampering and adds minimal build integrity guarantees

Level 3

Hardens the infrastructure against attacks, more trust integrated into complex systems

Level 4

The highest assurances of build integrity and measures for dependency management in place

[7]

WHO ARE WE

SPLYNTER

We are a small team of extremely driven security enthusiasts.

Michael
CISM, CISSP, OSCP, CEH & DPO

Mathijs
IoT expert, CEH, master of engineering

Sarah
Adds a human-centric approach to security, master in computer science

Glenn
Crypto expert, master in electrical engineering

VISION

A world where (complex) environments are cyber secure by default, not by exception.

MISSION

At Splynter our mission is to holistically improve the cyber security posture of companies around the globe through a vendor-independent and risk-based approach, combining high-quality security architecture with in-depth technical cyber security expertise.

VALUES

Embrace expertise

Take responsibility

QUESTIONS?

combat your adversaries with structure.

michael.boeynaems@splynter.be

REFERENCES

[1] https://azure.microsoft.com/en-us/overview/devops-vs-agile/

[2] Lean security, https://www.lean-security.org/

[3] https://digital.ai/devops-tools-periodic-table

[4] https://www.microfocus.com/en-us/what-is/

[5] https://www.ibm.com/cloud/learn/infrastructure-as-code

[6]https://www.linkedin.com/pulse/secrets-management-why-needed-bharat-kandanoor

[7] https://slsa.dev/spec/v0.1/threats

[8] https://owaspsamm.org