A PROGRAM-BASED APPROACH TO SECURING SOFTWARE DEVELOPMENT

Stan Letarte, CISM

vCISO and Senior Security Strategist, GreyCastle Security

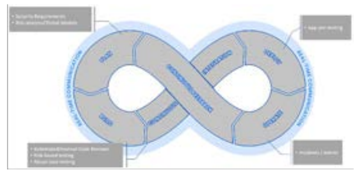
sletarte [at] GreyCastleSecurity [dot] com

OUTLINE

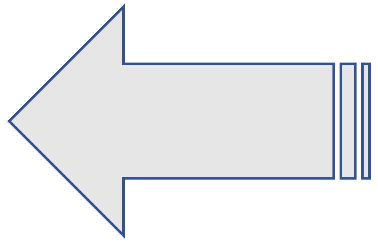
Problem Space

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Software Security



Shift Left



Getting Started

IS THERE A PROBLEM?

"Now Every Company Is a Software Company." – David Kirkpatrick in FORBES 188.11

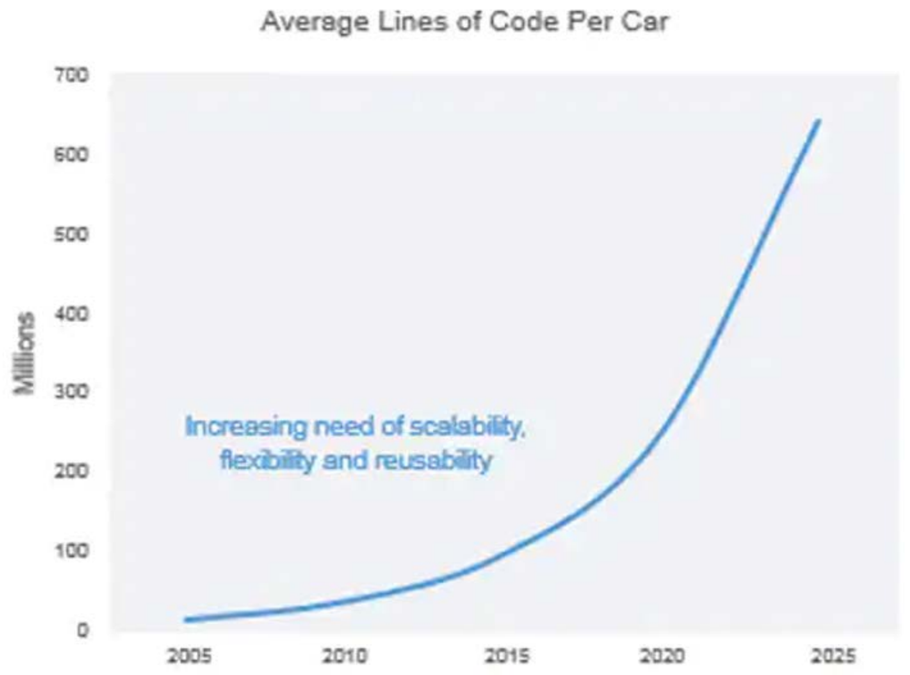
The software and technology industry is one of the fastest growing industries.

Supposition: Software Security has not kept pace with the rest of information security.

Many software developers are not security-trained; likewise, many security professionals are not software-conversant.

Focus tends to be on securing the app, not securing the process that built the app.

Illustration: the connected car has over 300M lines of code by 2020* (and will have more LOC than some aircraft by 2025!)



* https://community.nxp.com/pwmxy87654/attachments/pwmxy87654/connects/183/1/AMF-AUT-T2701.pdf

QUANTIFYING THE PROBLEM

- 76% of apps have at least one flaw; 24% of apps contain high severity flaws.*
- Although 30% of the flaws exist in the "homegrown app", nearly 70% exist in third-party libraries used by the app.*
- Black Hats know that there is better than a 1 in 3 chance of a data breach if they can find and exploit a vulnerability in an app.*
- 90 percent of apps aren't tested for vulnerabilities during their development and quality assurance stages, and even more go unprotected during production.**
- SecurityWeek claims that software supply chain attacks tripled in 2021. ***

*	Veracode – the state of software security, volume 11
**	Contrast Security
***	SecurityWeek, Jan 20, 2022

FURTHER EVIDENCE

(What supply chain hacks?!)

2014: **Target** .. Injection of POS skimmer code into the POS codebase

2021: **SolarWinds** .. Malicious code inserted into Orion

2021: **Codecov** .. Bash uploader compromised – hacker access to the CI processes of its customers

2021: **ua-parser-js** .. Modified and placed on NPM to push Crypto Mining malware

Three common attack vectors:

- Vulnerabilities in packages used (particularly Open Source packages)
  - Exploiting known vulnerabilities
  - Poisoning the package
- Compromising pipeline tools

ENTER: SOFTWARE SECURITY

- Software Security, a subset of Cybersecurity, is a largely unregulated*, often unguided frontier, but represents a huge opportunity in which to catch flaws before they are deployed and become vulnerabilities.
- Software Security needs both destructive and constructive activities, and occurs throughout the SDLC.
  - Destructive activities: breaking software through attacks and exploits (black hat -- offense).
  - Constructive activities: design, defense, and security functionality (white hat -- defense).

  Both hats are necessary!
- Some elements of software security are "programming-specific", but many are not, and require a Program-based approach.

The goal of software security isn't to write applications perfectly the first time, but to remediate the flaws in a comprehensive and timely manner.

* *The Federal government is finally interested in S/SDLC! See: Executive Order 14028 (Sec 4, May 12, 2021). The response with NIST publication 800-218 (2/3/2022) is indicative that this is going to be a "wild ride" area of infosec during the future years.*

WHY IS SOFTWARE SECURITY DIFFERENT?

Classic security programs align work priorities with risk, building roadmaps that answer:

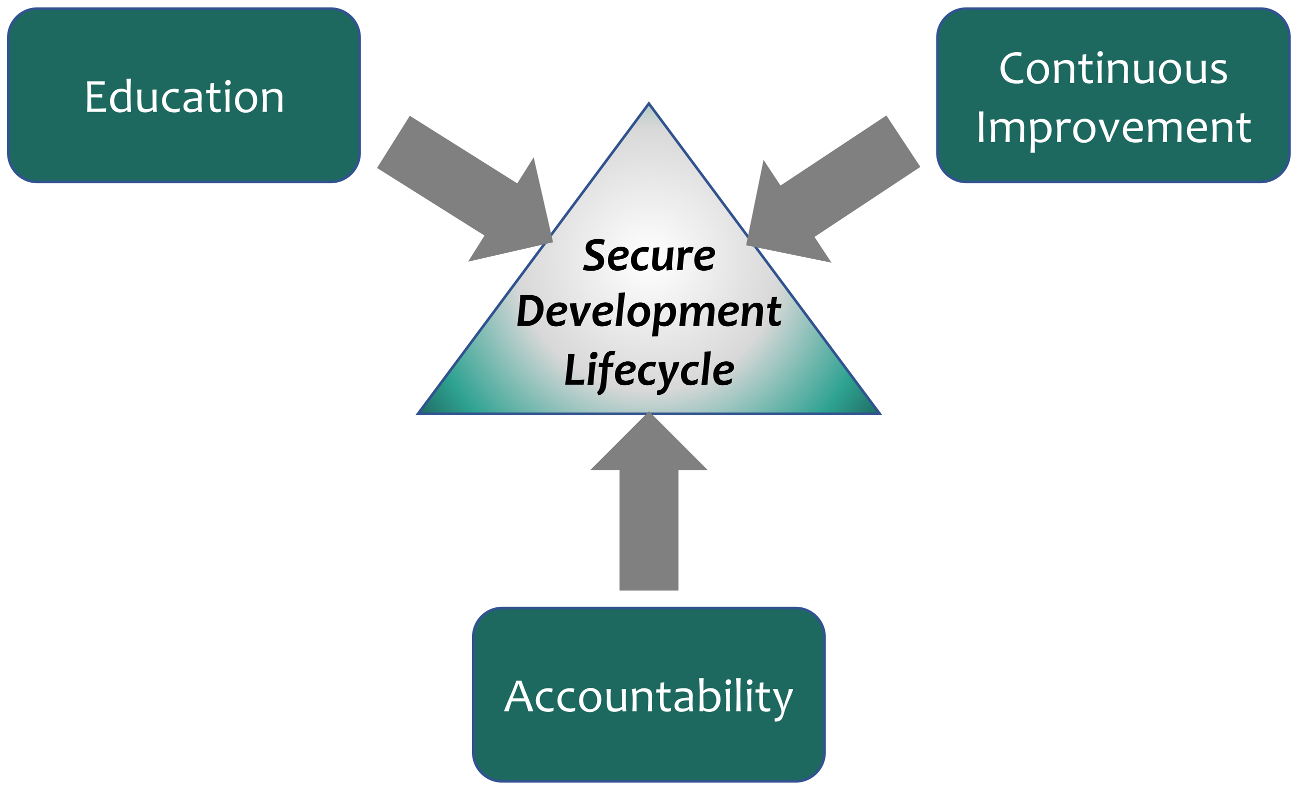
- What needs to be done
- How much should be done
- In what order should we proceed

Language/Stack/discipline make for a massive variance in risk!

Software Security could benefit from a risk-based approach, *yet there is no universal control framework that addresses risk-based software security with adequate depth to become actionable in the SDLC.*

DRIVERS FOR SECURING DEVELOPMENT

There are three disciplines that can help secure our software development practices:



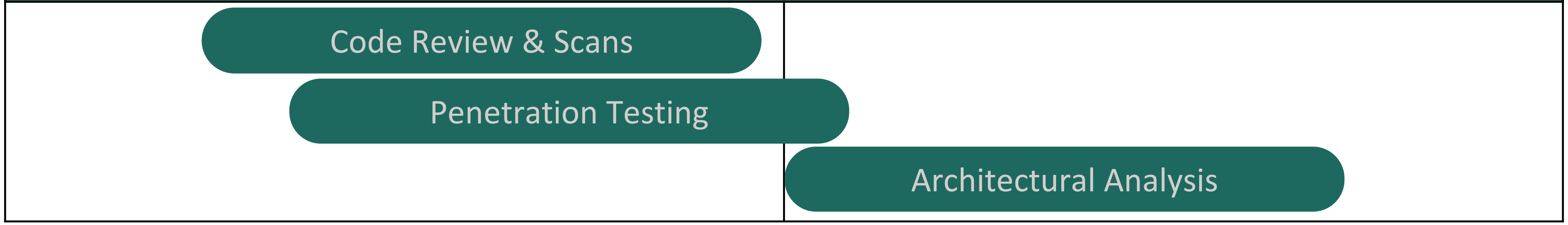
DOESN'T SECURITY JUST START WITH CODE?

Short answer: NO!

BUGS ≠ FLAWS!

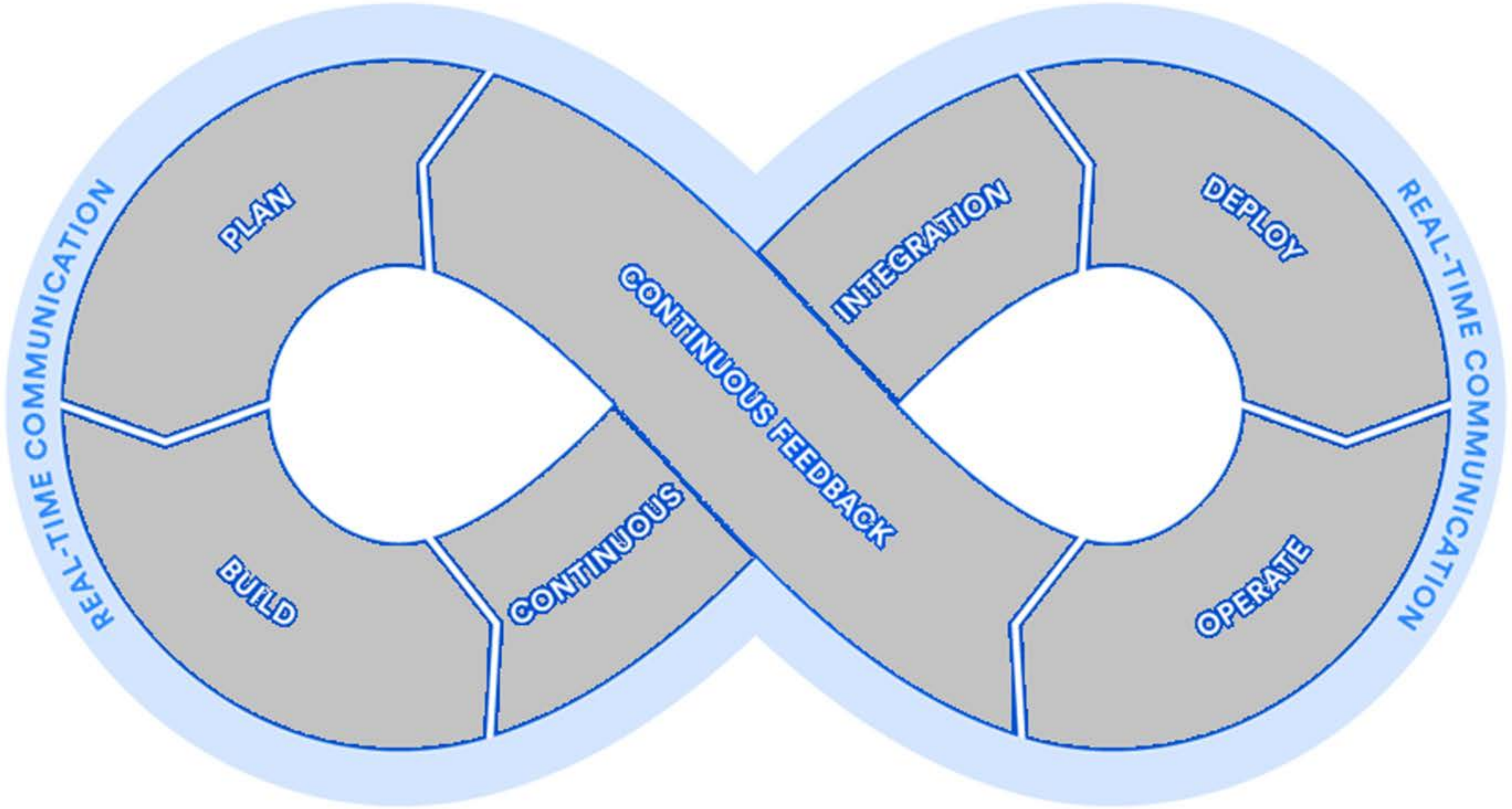
In most studies, **bugs** and **flaws** divide the defect space approximately 50/50.*

Bugs	Flaws
Found in implementations	Found in Design/Architecture
• Examples:	• Examples
• Cross-site scripting	• Weak/missing security control
• Buffer Overflow	
Testing	



* Jim DelGrosso and Gary McGraw: https://searchsecurity.techtarget.com/opinion/Opinion-Software-insecurity-software-flaws-in-application-architecture

SO … WHY WAIT FOR CODE TO START SECURING?



SHIFTING THE SECURITY FOCUS

A Security Manifesto, in the spirit of Agile

- **Rely on developers and testers** *more than security specialists*.
- **Secure while we work** *more than after we're done*.
- **Implement features securely** *more than adding on security features*.
- **Mitigate risks** *more than fixing bugs*.

Although this speaks to a culture change, we often need a program (goals/actions/measures/rewards) to get the culture to change.

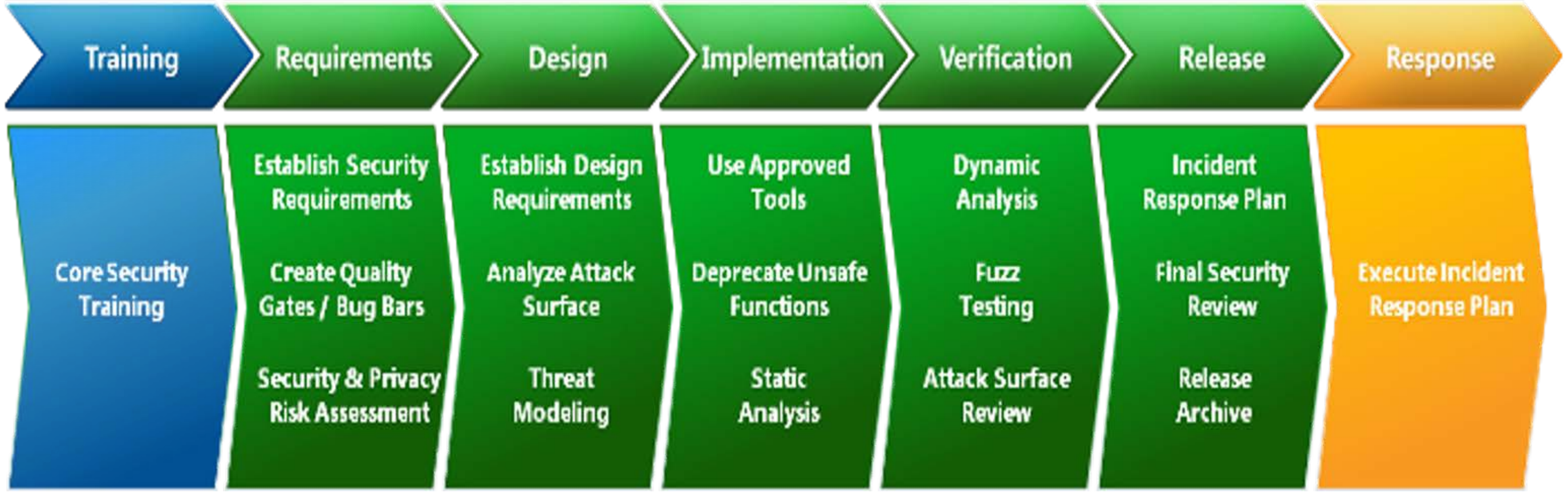
FRAMEWORK-BASED APPROACHES

Survey of tried & true frameworks

- Microsoft SDL
- OWASP OpenSAMM
- PCI Secure Software Lifecycle
- NIST 800-218 / Secure Software Development Framework
- BSIMM

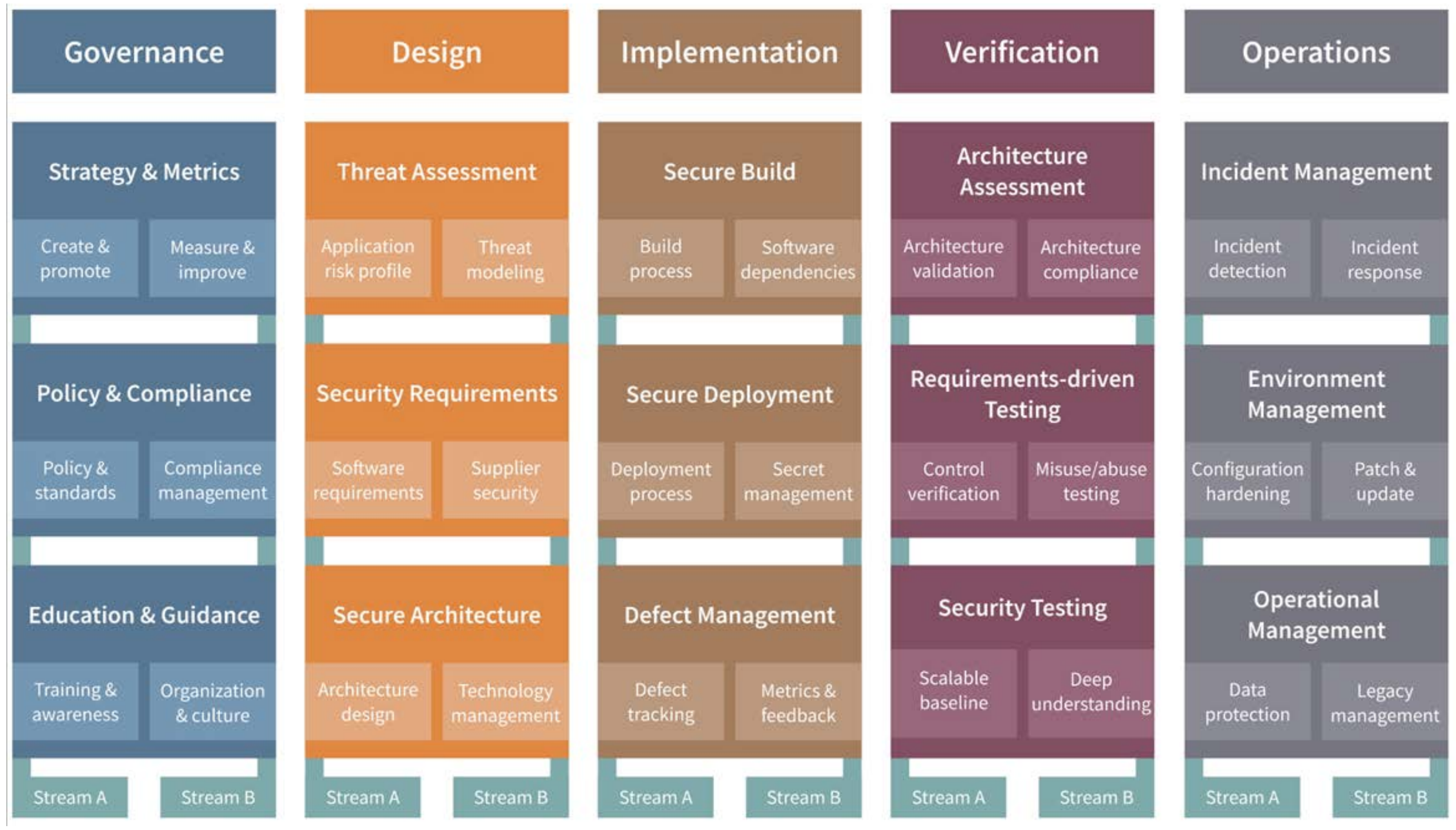
MICROSOFT SECURITY DEVELOPMENT LIFECYCLE (SDL)

- One of the first! In response to Windows XP, the SDL became mandatory for most Microsoft products in 2004.
- Heavy emphasis on Microsoft-specific technologies / culture



OWASP OPENSAMM

- Software Assessment Maturity Model
- Somewhat prescriptive

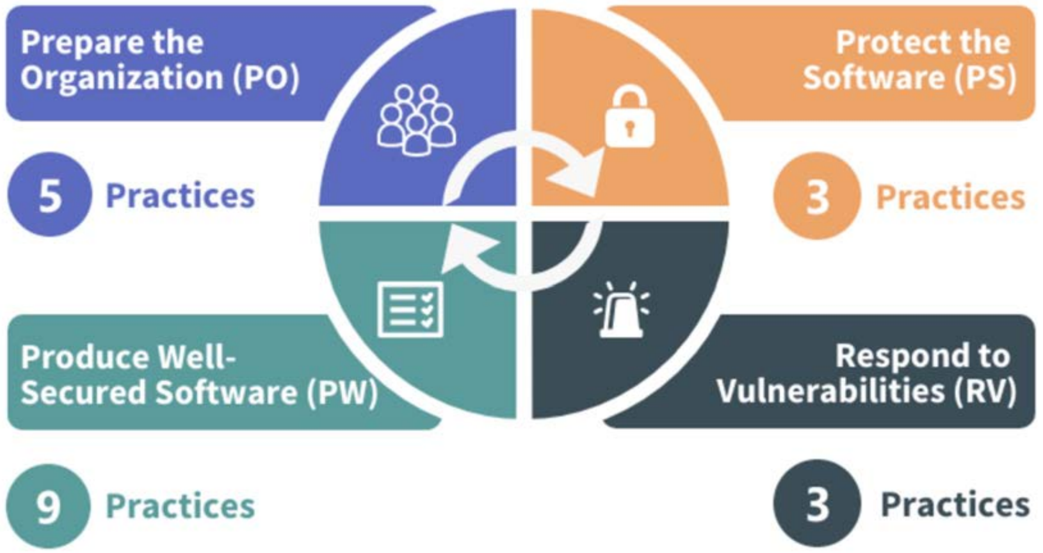


PCI SECURE SOFTWARE DEVELOPMENT LIFECYCLE (SSDL)

- This is one half of the PCI standards for securing payment apps
- Specifies the processes by which an organization should secure its software
- Specific to credit card security, but can provide good insights beyond

NIST SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SP 800-218, 2022)

- The NIST response to Executive Order 14028 (in response to SolarWinds attacks on Federal agencies)
- Highly influenced by NIST CSF and BSA.org Framework for Secure Software
- Four Processes:

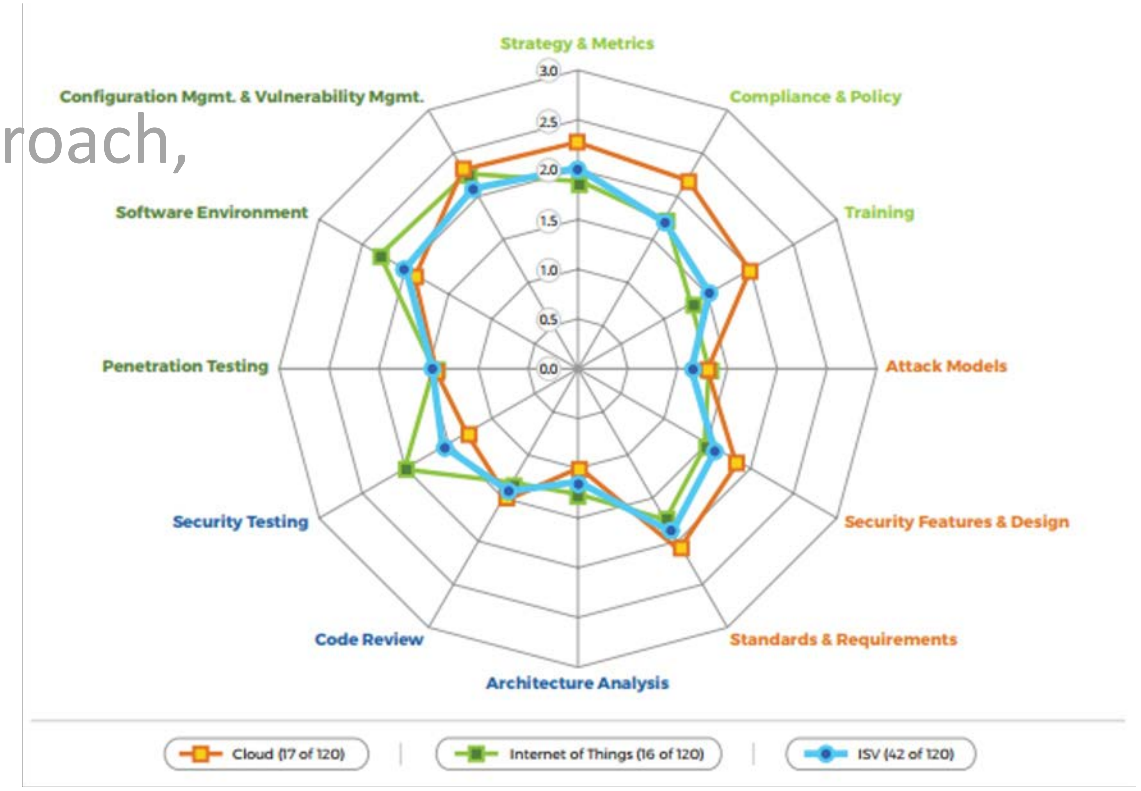

Prepare the Organization (PO)
5 Practices
Protect the Software (PS)
3 Practices
Produce Well-Secured Software (PW)
9 Practices
Respond to Vulnerabilities (RV)
3 Practices

- Has an extensive list of references to other frameworks and resources

Graphic credit: Chainguard, Inc

BUILDING SECURITY IN MATURITY MODEL (BSIMM)

- Measures maturity of SDLC practices
- Provides industry-specific maturity benchmarks
- Does not attempt to prescribe your approach, but tells what others are doing
- Framework has matured over 11 years



BSIMM IN MORE DEPTH ..

Governance. Practices that help organize, manage, and measure a software security initiative. Staff development is also a central governance practice.

Intelligence. Practices that result in collections of corporate knowledge used in carrying out software security activities throughout the organization. Collections include both proactive security guidance and organizational threat modeling.

SSDL Touchpoints. Practices associated with analysis and assurance of particular software development artifacts and processes. All software security methodologies include these practices.

Deployment. Practices that interface with traditional network security and software maintenance organizations. Software configuration, maintenance, and other environment issues have direct impact on software security.

- Each of the four domains has three Practices
- Each Practice has a varying number of Activities
- Activities have a maturity scale (1 through 3)
- Activities that are no longer observed get dropped from the benchmark

GETTING STARTED

- Recommend: use BSIMM to educate / assess your team(s)
- If fully assessing, compare with the benchmarks of other best-in-class companies
- Create a “starting list” of activities to consider as starting points:
  - Look for ways to shift left: where do you first start “securing” ?
  - Begin with the 12 most common activities that other companies do: ask “why not”
  - To start: eliminate or modify activities that depend on advanced organization/concepts, such as:
    - SSG
    - SSG Satellite
- Perform stack-ranking to determine a prioritized order of activity adoption
- Consider Policy mechanisms to make it stick
- Continuous improvement: assess & reprioritize annually

Q&A