OWASP
AppSec Europe
London 2nd-6th July 2018

Secure Software Dev. Framework

Towards an SDL for all SDLCs

Damilare D. Fagbemi

My Friends & Co-Authors

- Tony Martin @ Intel Corp
- Raghudeep Kannavara @ Intel Corp
- Priti Shrivastav @ Intel Corp
- Brook Schoenfield @ McAfee

Secure Software Dev. Framework

Damilare D. Fagbemi

A little about me…

- Software Security Architect @ Intel Corp
- Platforms: Web, Mobile, Windows, Linux
- OWASP Nigeria Chapter Founder
- Blog @ https://edgeofus.com
- Past Life in Tech:
  - Co-Founder, Web consultancy
  - Software Engineer
- Working on my first book… wish me Luck… really ☺

Secure Software Dev. Framework

Damilare D. Fagbemi

Disclaimer!!!

I speak for my self and not Intel Corporation.

Target Audience and Expected Outcomes

Audience:

- Software Engineers/ Developers
- Software Architects
- Engineering Managers
- Security Practitioners (Engineers, Architects, Researchers, CISOs)
- Anyone and everyone who’s interested!

Expected outcomes:

- An understanding of the purpose the SDL
- Insight to ways the SDL hurts
- Learn the benefits of using SSDF’s lean SDL techniques
- Learn how to map SSDF to product development

Secure Software Dev. Framework

Damilare D. Fagbemi

Devices, Platforms and Software

We are facing a future of unbounded complexity…
There are already many more computing devices in the world than there are people. In a few more years, their number will climb into the trillions.

- Peter Lucas, Joe Bailey, Mickey McManus, Trillions

Software is like entropy: It is difficult to grasp, weighs nothing, and obeys the Second Law of Thermodynamics; i.e., it always increases.

– Norman Augustine

Secure Software Dev. Framework

Damilare D. Fagbemi

Software anyone?

Secure Software Dev. Framework

Damilare D. Fagbemi

Software complexity >>>>>>>

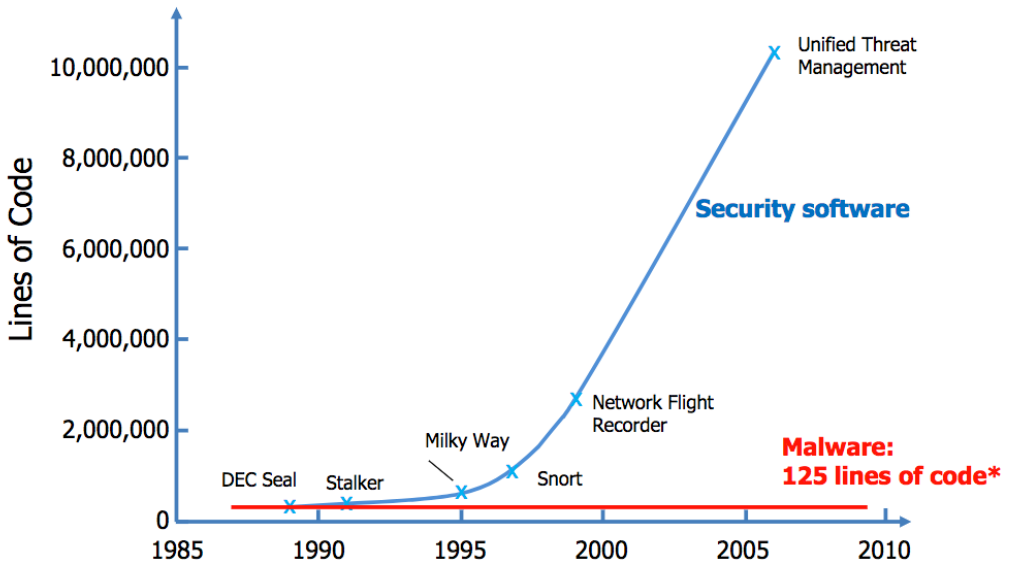
Product	~ Lines of Code
Linux Kernel 2.2 (99)	1,800,000
Linux Kernel 2.6 (03)	5,929,000
Linux Kernel 3.1 - 2013	15,800,000
Linux Kernel 4.14 - 2017	23,200,000
Average Car - 2010 [2]	10,000,000
Firefox - March 2018 [3]	36,897,000
F150 Ford Truck - 2016 [4]	150,000,000

Secure Software Dev. Framework

Damilare D. Fagbemi

Security complexity >>>>>>>

DARPA The Problem: Not Convergent

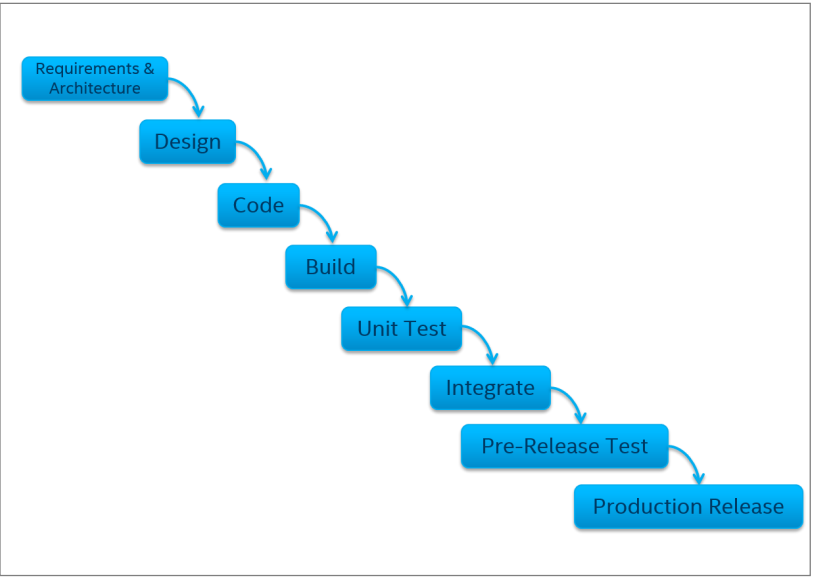


* Public sources of malware averaged over 9,000 samples (collection of exploits, worms, botnets, viruses, DoS tools)

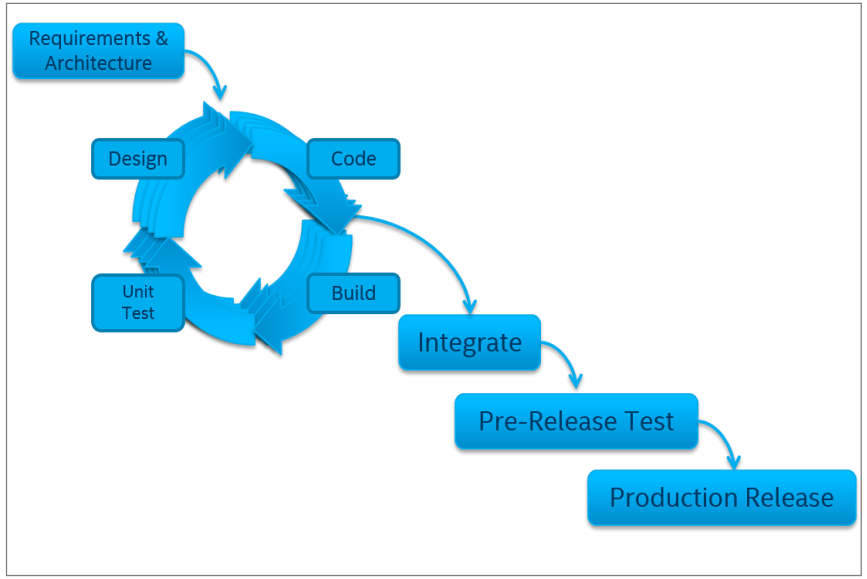
Secure Software Dev. Framework

Damilare D. Fagbemi

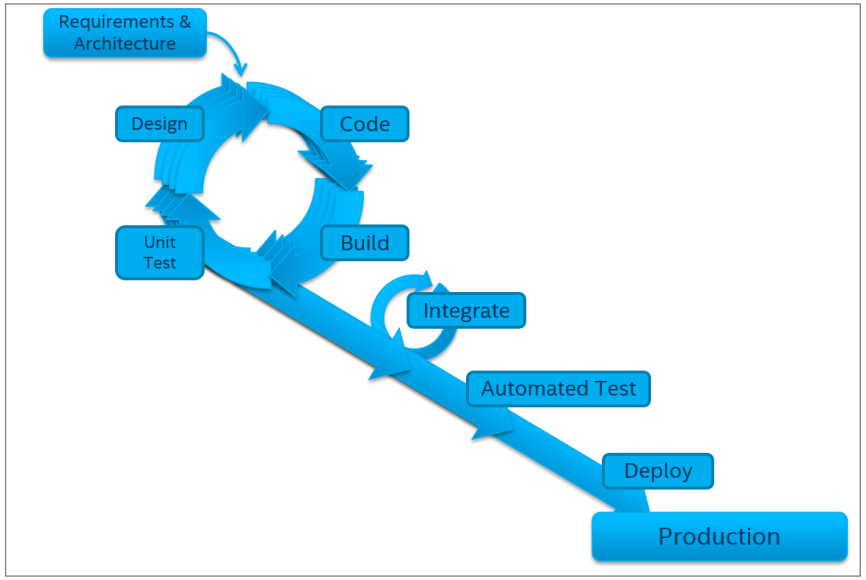
How We Build Software > SDLCs



Waterfall



Iterative

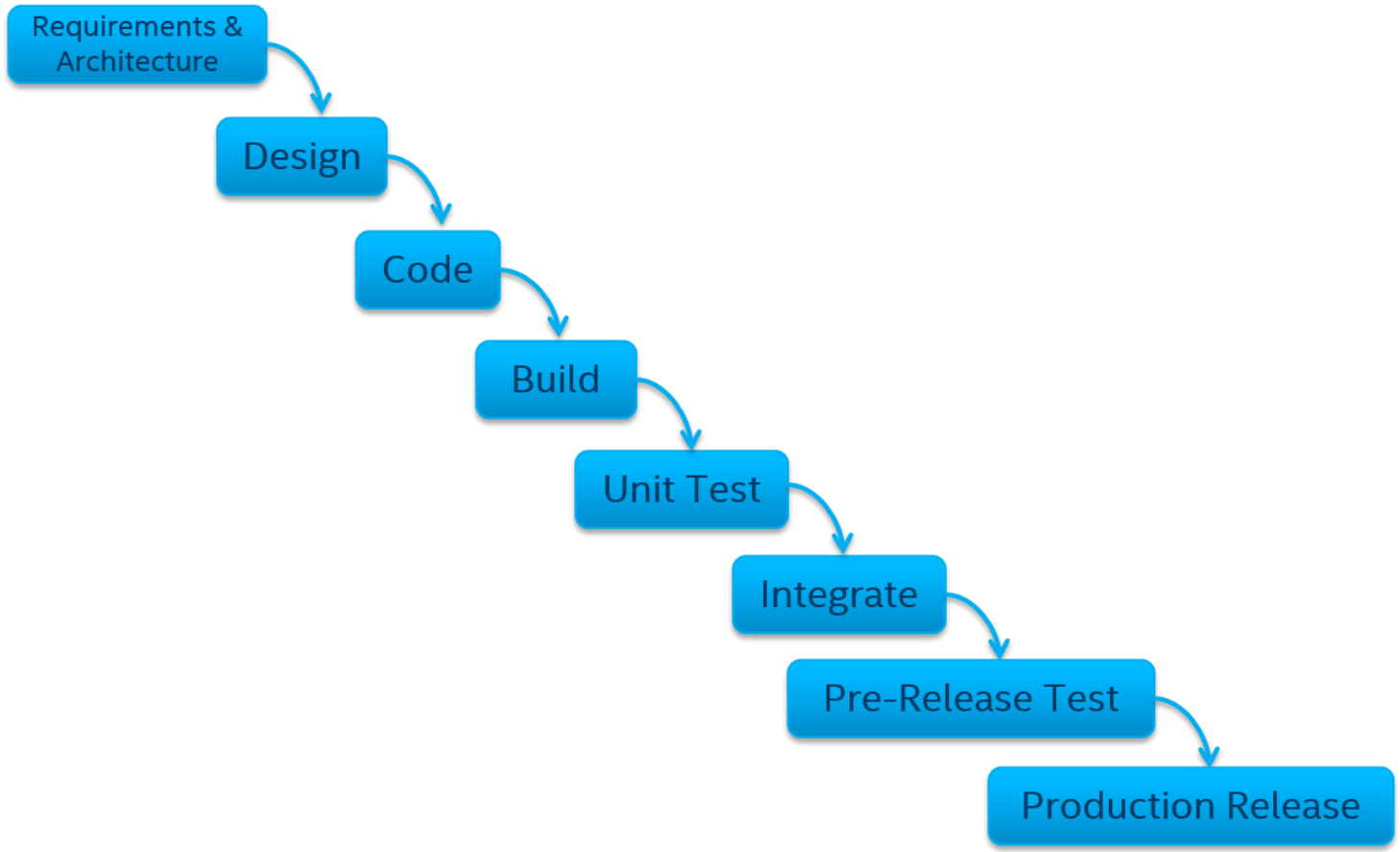


Continuous Deployment

Secure Software Dev. Framework

Damilare D. Fagbemi

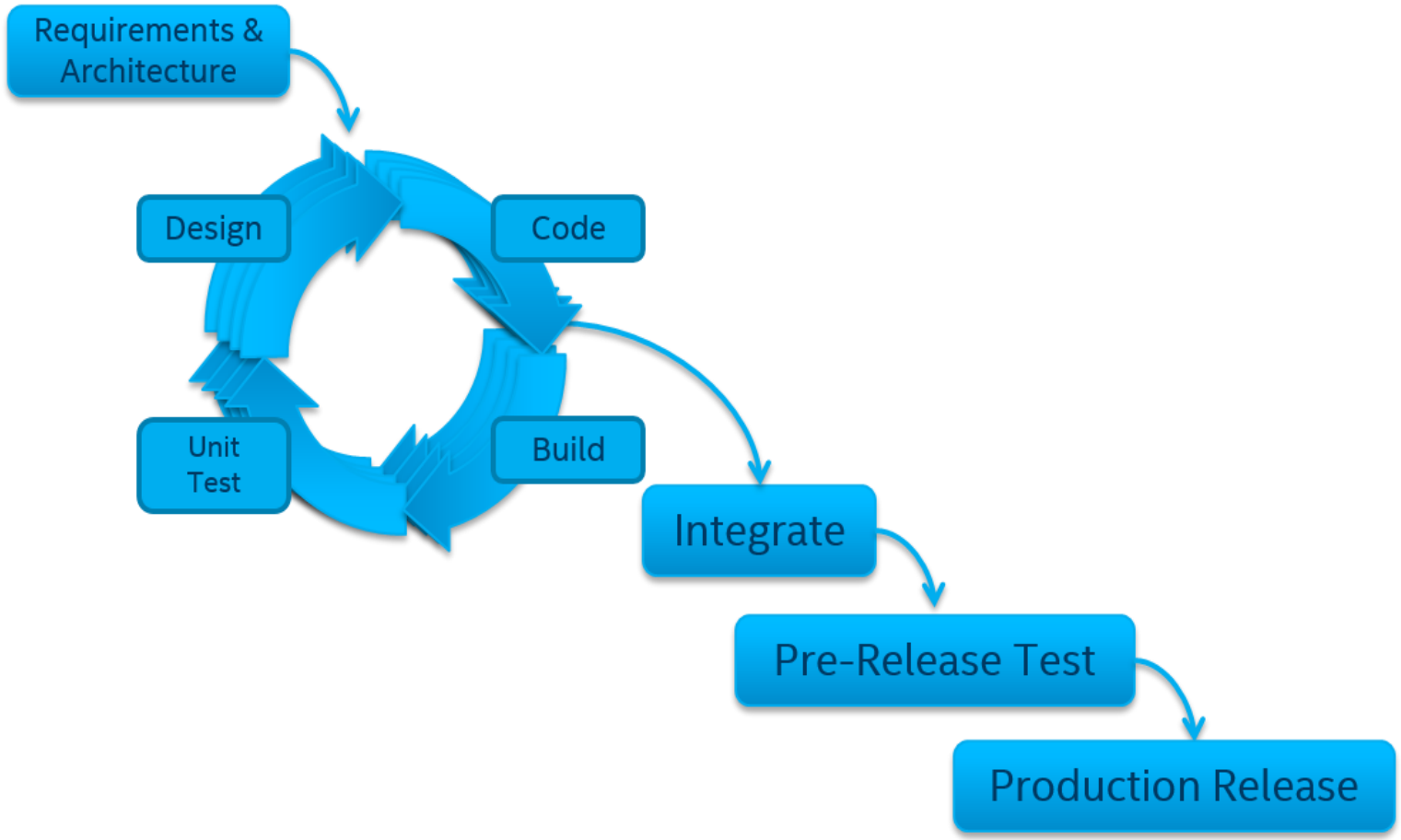
The Waterfall Model



Secure Software Dev. Framework

Damilare D. Fagbemi

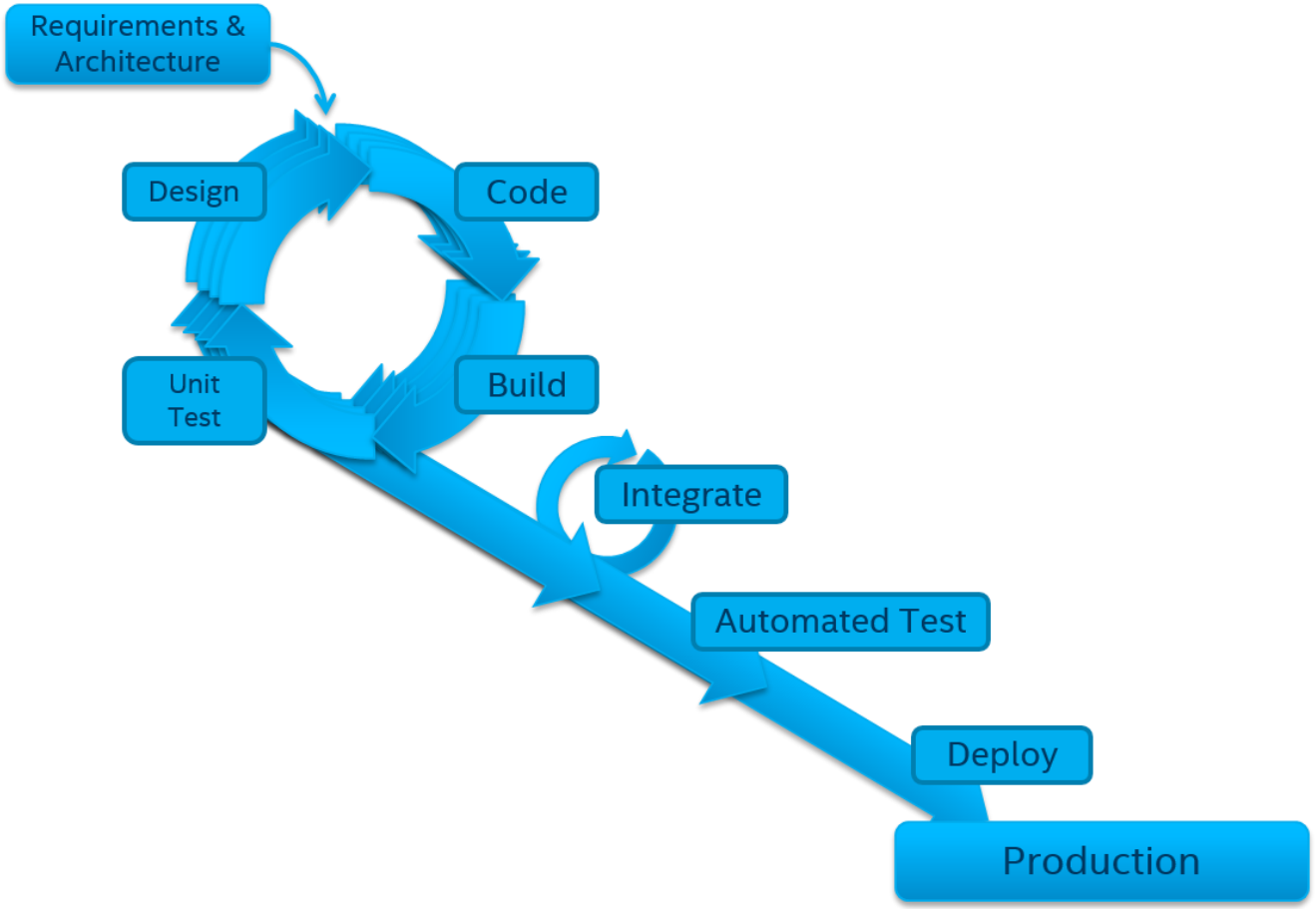
The Iterative Model



Secure Software Dev. Framework

Damilare D. Fagbemi

The Continuous Deployment Model



Secure Software Dev. Framework

Damilare D. Fagbemi

The SDL… arrived 2004ish

1. TRAINING	2. REQUIREMENTS	3. DESIGN	4. IMPLEMENTATION	5. VERIFICATION	6. RELEASE	7. RESPONSE
1. Core Security Training	2. Establish Security Requirements	5. Establish Design Requirements	8. Use Approved Tools	11. Perform Dynamic Analysis	14. Create an Incident Response Plan	Execute Incident Response Plan
	3. Create Quality Gates/Bug Bars	6. Perform Attack Surface Analysis/ Reduction	9. Deprecate Unsafe Functions	12. Perform Fuzz Testing	15. Conduct Final Security Review	
	4. Perform Security and Privacy Risk Assessments	7. Use Threat Modeling	10. Perform Static Analysis	13. Conduct Attack Surface Review	16. Certify Release and Archive	

The Pains of The SDL

Secure Software Dev. Framework

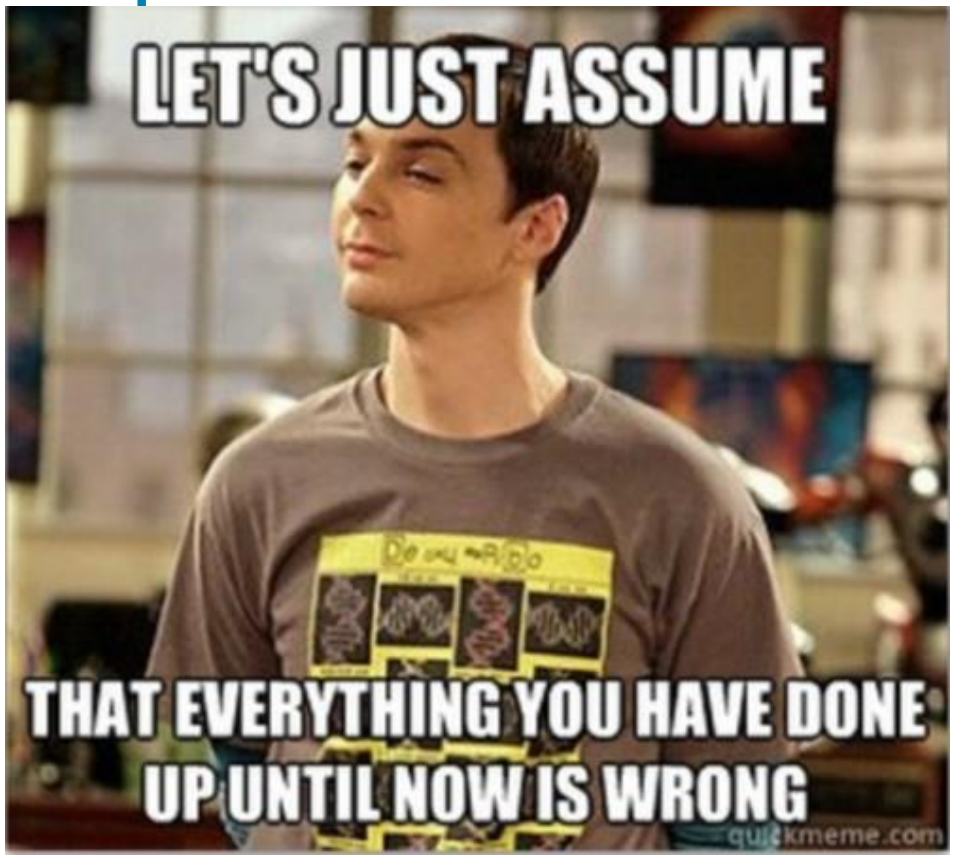
Damilare D. Fagbemi

Pain 1: Devs don’t fully understand their SDLC

Secure Software Dev. Framework

Damilare D. Fagbemi

Pain 2: Security Engineers constrain the SDL to their narrow/ specific SDLC understanding

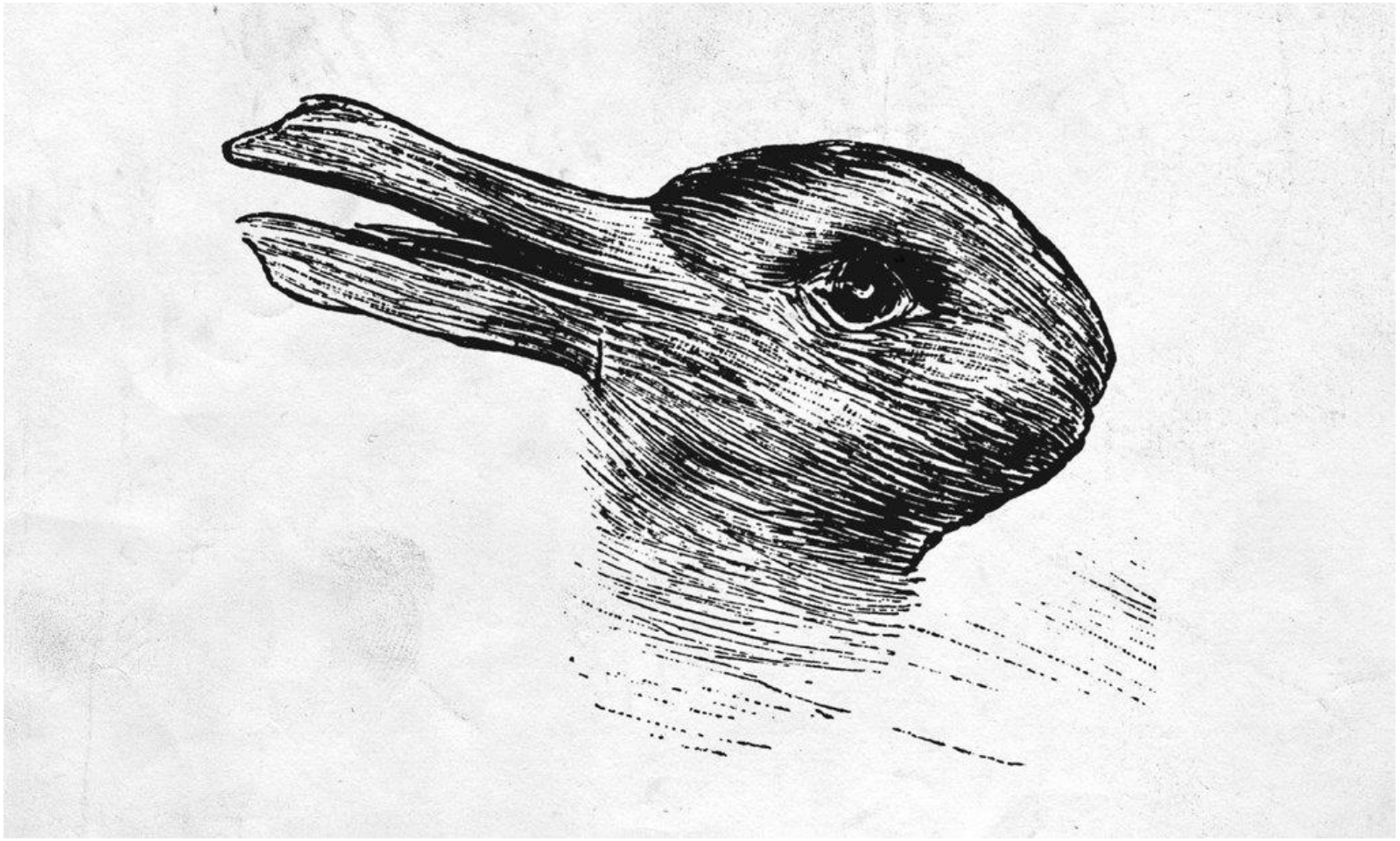


Secure Software Dev. Framework

Damilare D. Fagbemi

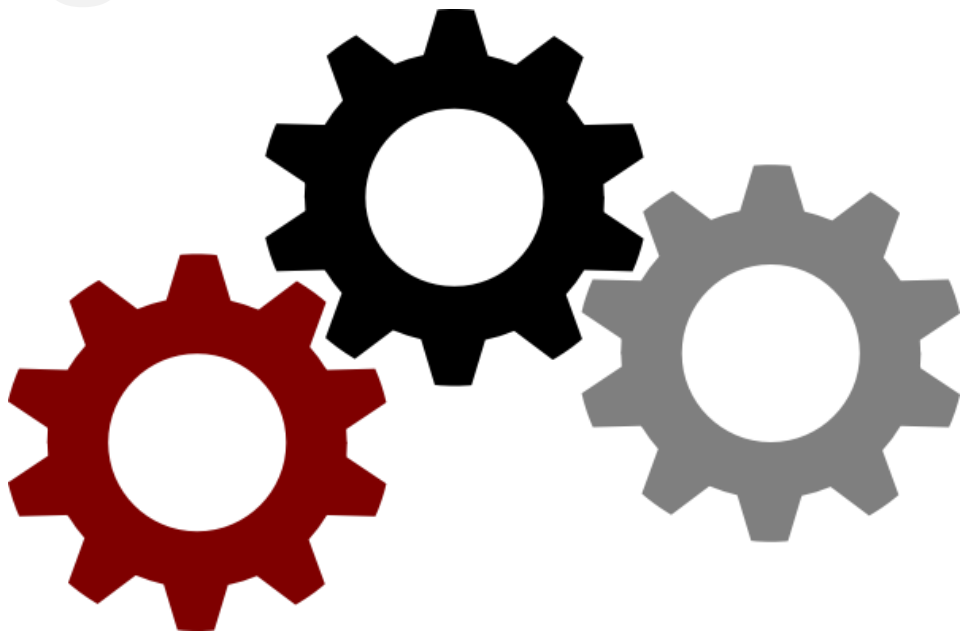
Pain 3: SDL <-> Product Development mapping unclear

Pain 4: Ambiguous Security Requirements



Pain 5: Inconsistency in SDL Execution

SSDF: Building an SDLC agnostic SDL



Secure Software Dev. Framework

Damilare D. Fagbemi

What is SSDF?

- SDLC agnostic
  - Adaptable
- Includes Re-visitation
- Minimal

Secure Software Dev. Framework

Damilare D. Fagbemi

SSDF Resolves SDL pain points…

- Developers don't fully understand their SDLC
- Security Engineers often constrain the SDL to their narrow/ specific SDLC understanding
- SDL <-> Product Development mapping unclear
- Ambiguous security requirements
- Inconsistency in SDL execution

Secure Software Dev. Framework

Damilare D. Fagbemi

Attributes of SSDF

Lean

Clear

Security Specific

Code Focused

Secure Software Dev. Framework

Damilare D. Fagbemi

SSDF's Activities

SSDF Activity	To Be Performed During…
Establish security and privacy constraints	Project/ feature initiation
Threat modeling & security architecture	Architecture
Secure design review; Manage 3rd party components*	Design
Code reviews; Static analysis	Coding
Vulnerability testing	Validation
Pre-release review	Release and Maintenance

Secure Software Dev. Framework

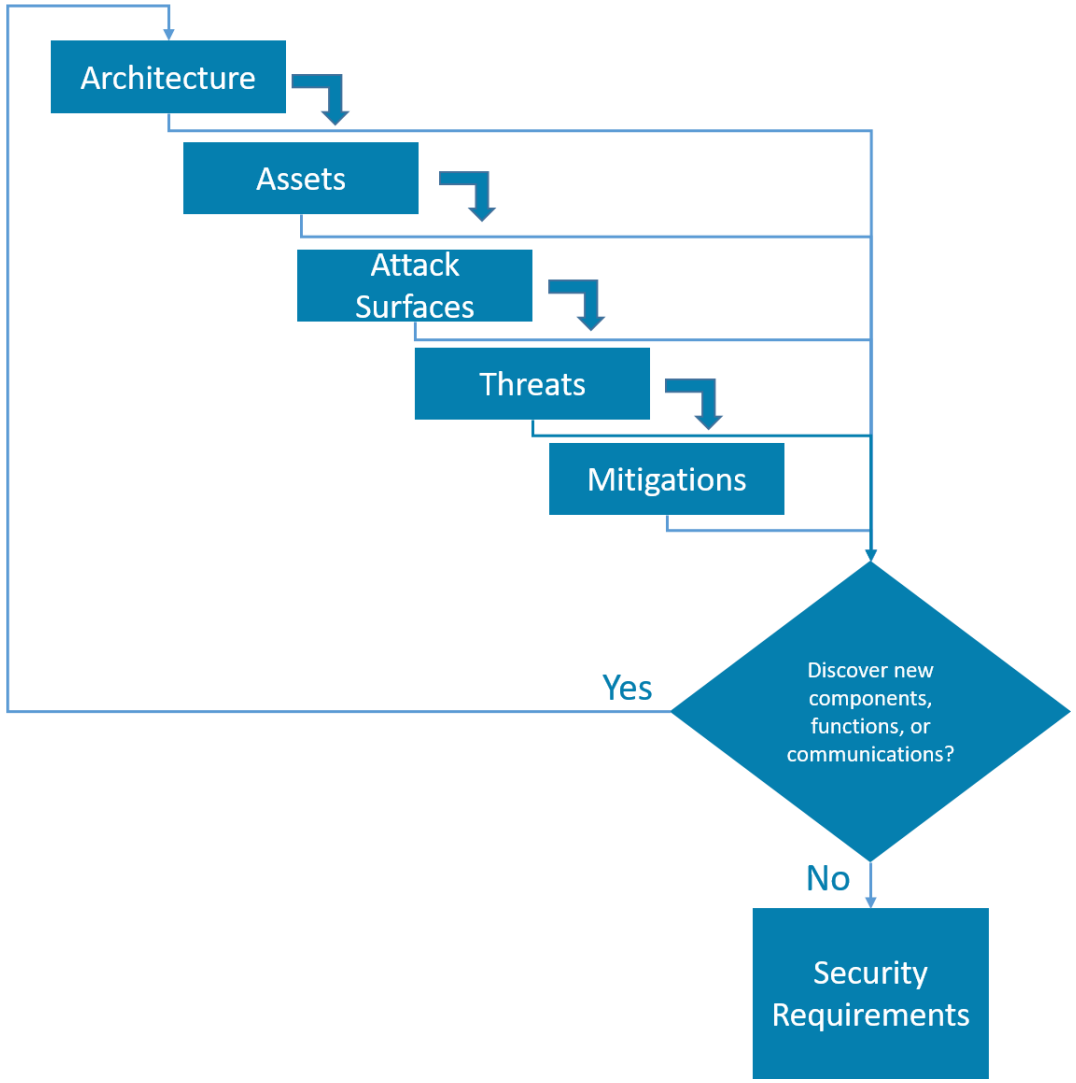
Damilare D. Fagbemi

SSDF: Establish Security and Privacy Constraints

Secure Software Dev. Framework

Damilare D. Fagbemi

SSDF: Threat Modeling & Security Architecture



Adapted from: Securing System, CRC Press, 2015

Secure Software Dev. Framework

Damilare D. Fagbemi

SSDF: Security Design Review

Secure Software Dev. Framework

Damilare D. Fagbemi

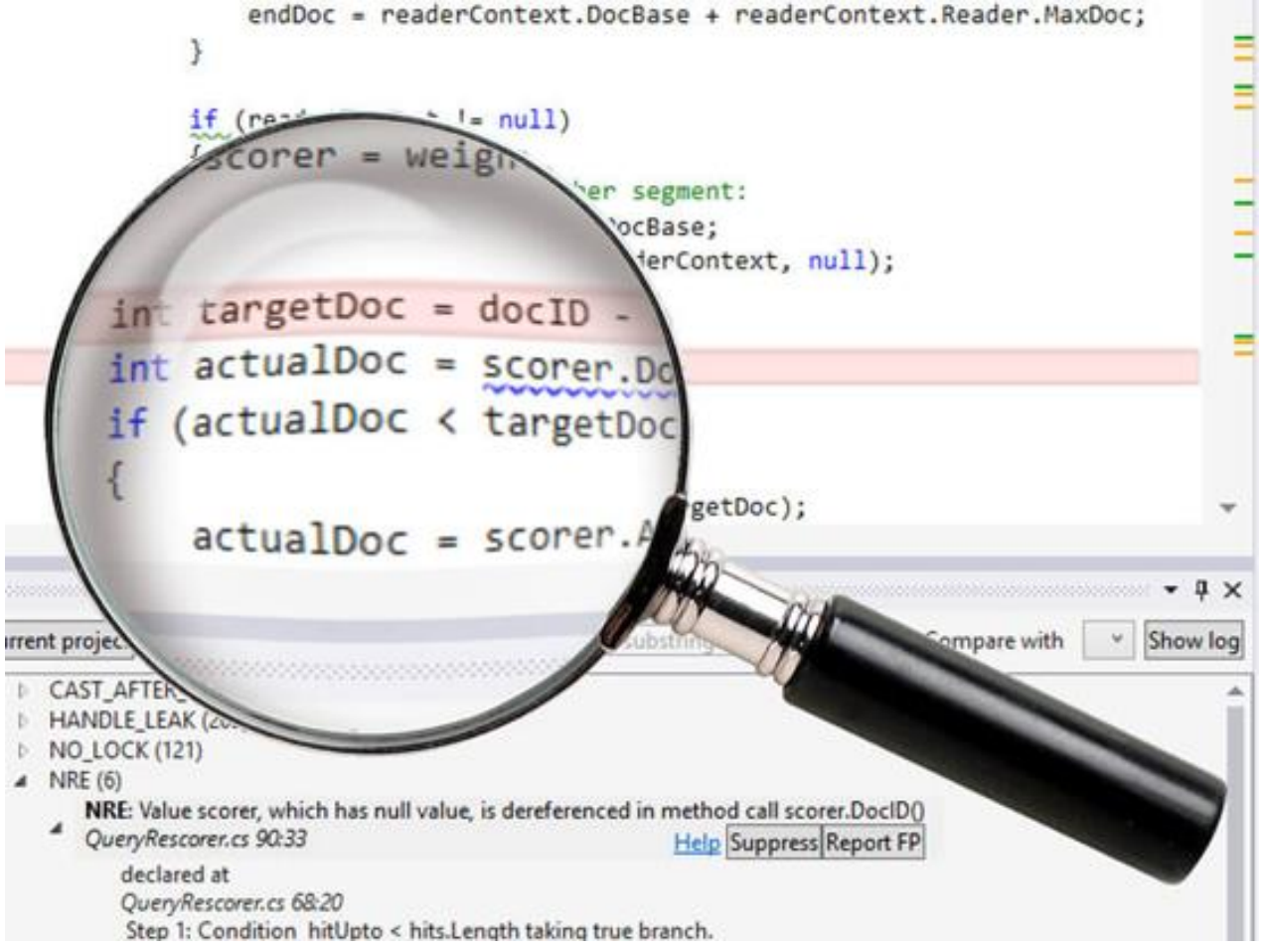
SSDF: Manage 3rd Party components

...

Secure Software Dev. Framework

Damilare D. Fagbemi

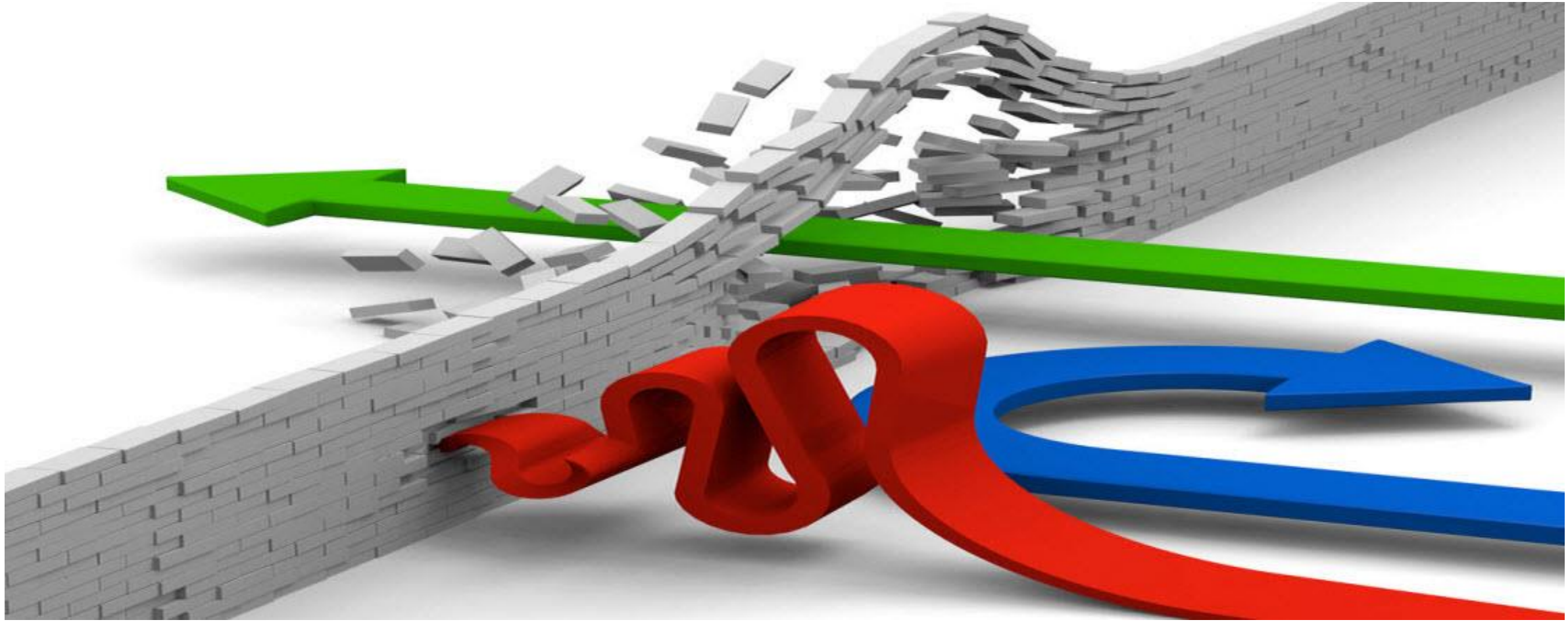
SSDF: Code Review; Static Analysis



Secure Software Dev. Framework

Damilare D. Fagbemi

SSDF: Vulnerability Testing



Secure Software Dev. Framework

Damilare D. Fagbemi

SSDF: Pre-release review

SSDF's Re-visitation Guidance

Problem Statement

- Project scope, features, architecture may change at anytime
  - Impacts SDL activities already done
- SSDF includes Guidance on when to revisit already done activities

Examples of Re-visitation Guidance in SSDF

Activity: Threat Modeling & Security Architecture

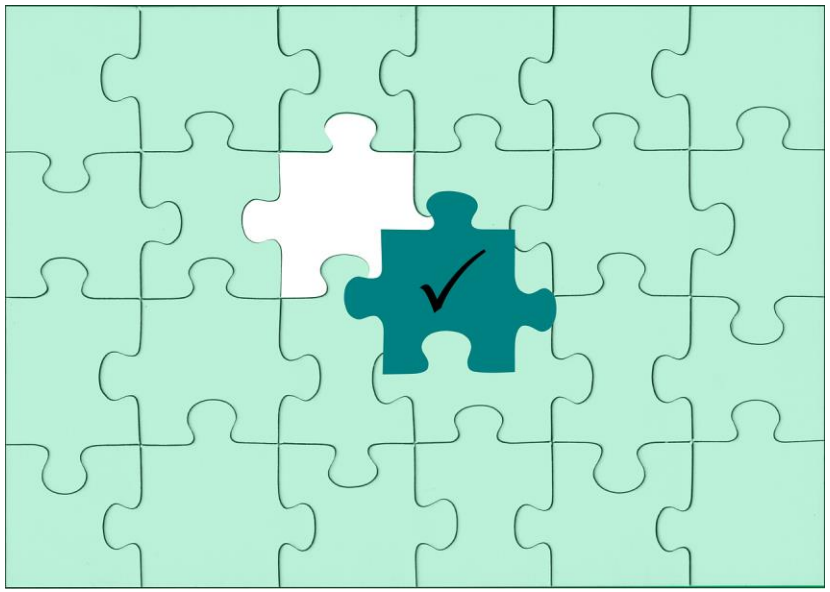
- This activity should be revisited:
  - New/modified requirements or design changes are identified which have the potential to impact data flows, overall architectural structure or mitigations
  - Whenever a new threat is identified that was not previosly accounted for. (e.g., security validation identifying a new threat, PSIRT incident,...)
  - If the overall product architecture has undergone incremental minor changes over time but has not been reviewed holistically in a year or more

Examples of Re-visitation Guidance in SSDF

Activity: Vulnerability Testing

- This activity should be revisited:
  - Anytime changes are made to product design or implementation, re-assess and determine if additional action needed
  - If implementation of a security feature (e.g. crypto,) or API/library/dependencies has changed
  - If the software interface or API definition has been modified in anyway
  - If a new security related defect is discovered (PSIRT), re-assess and determine if additional action needed

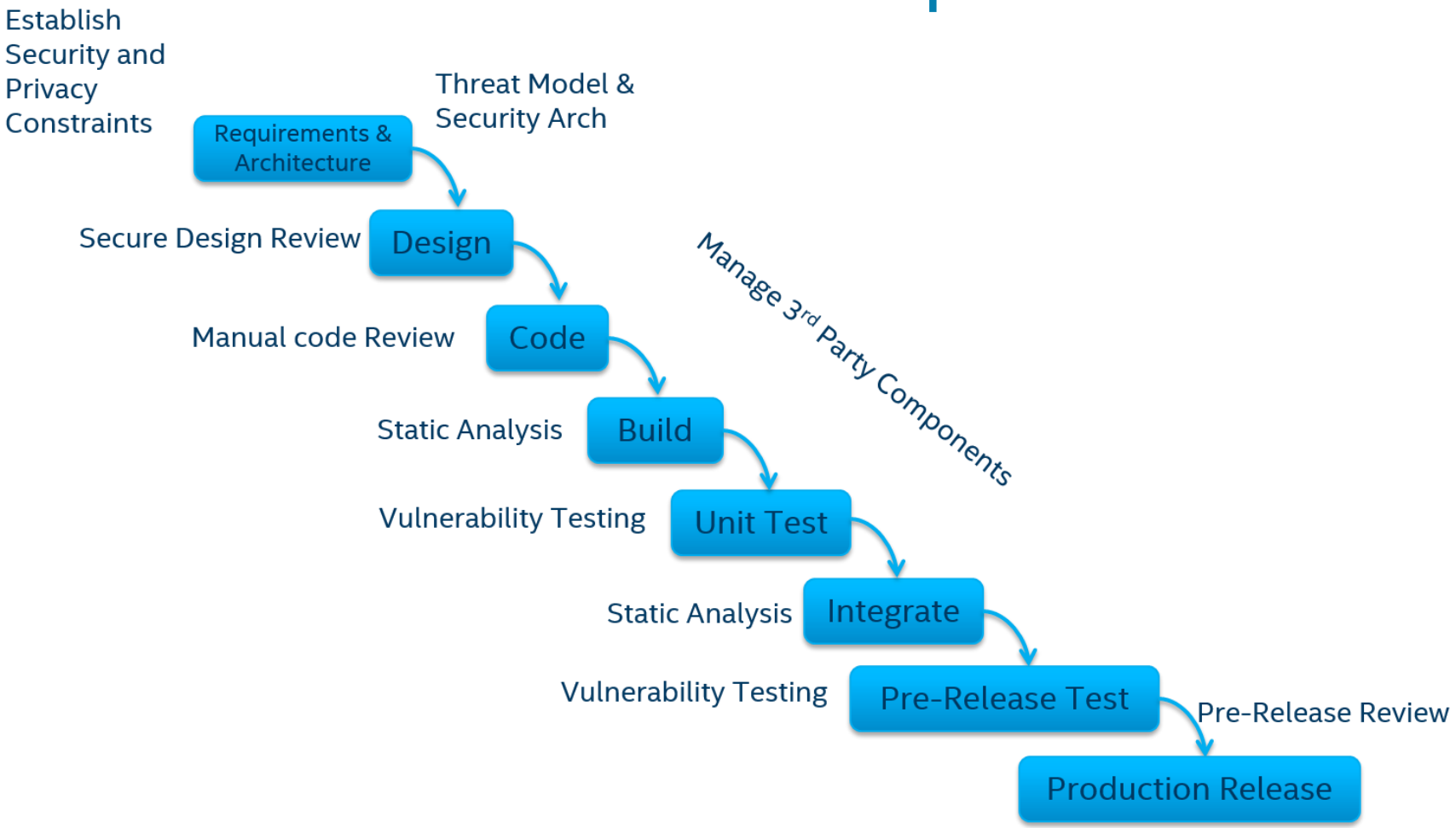
Mapping SSDF to 3 SDLCs



Secure Software Dev. Framework

Damilare D. Fagbemi

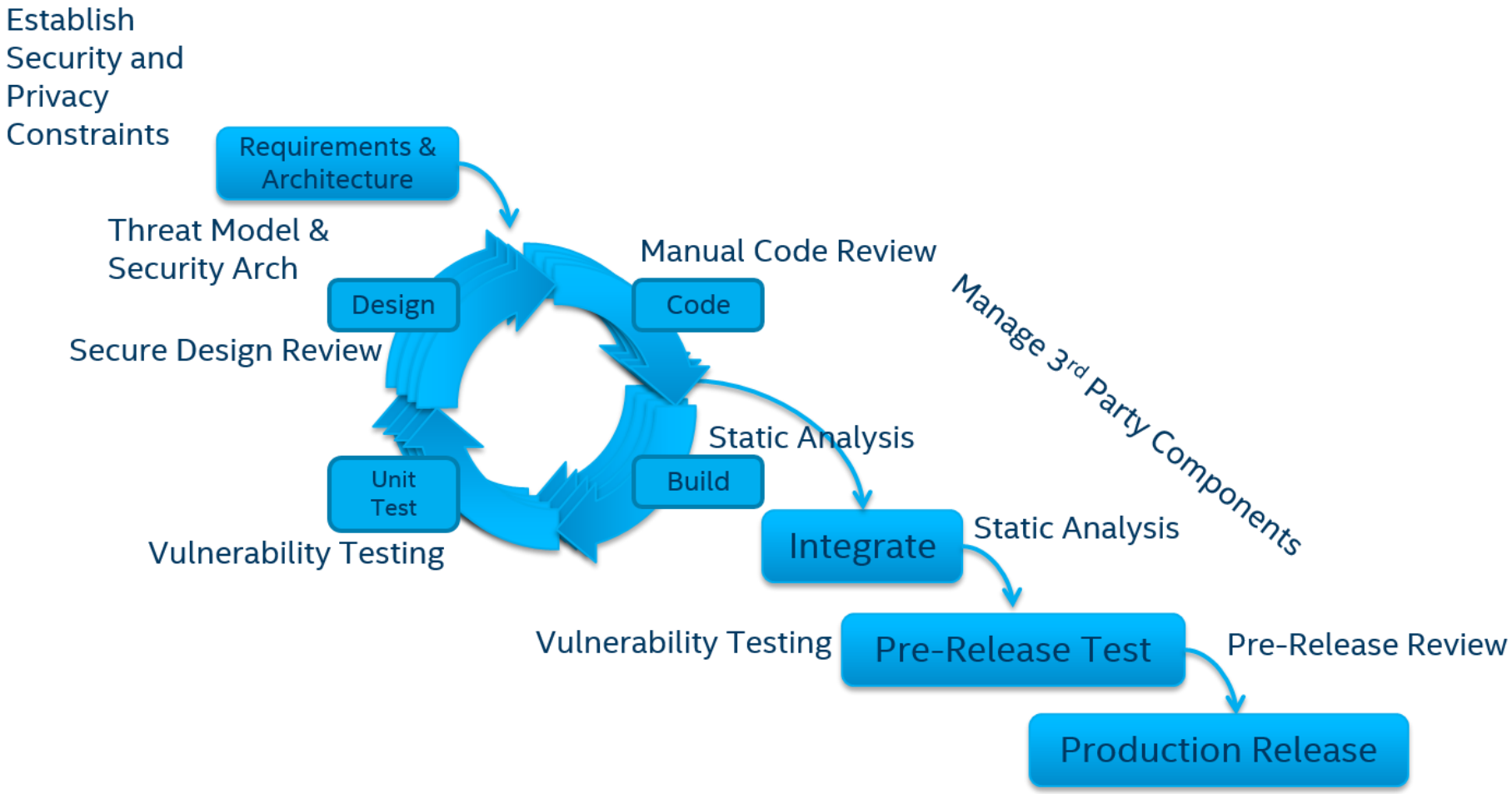
SSDF in Waterfall-based Development



Secure Software Dev. Framework

Damilare D. Fagbemi

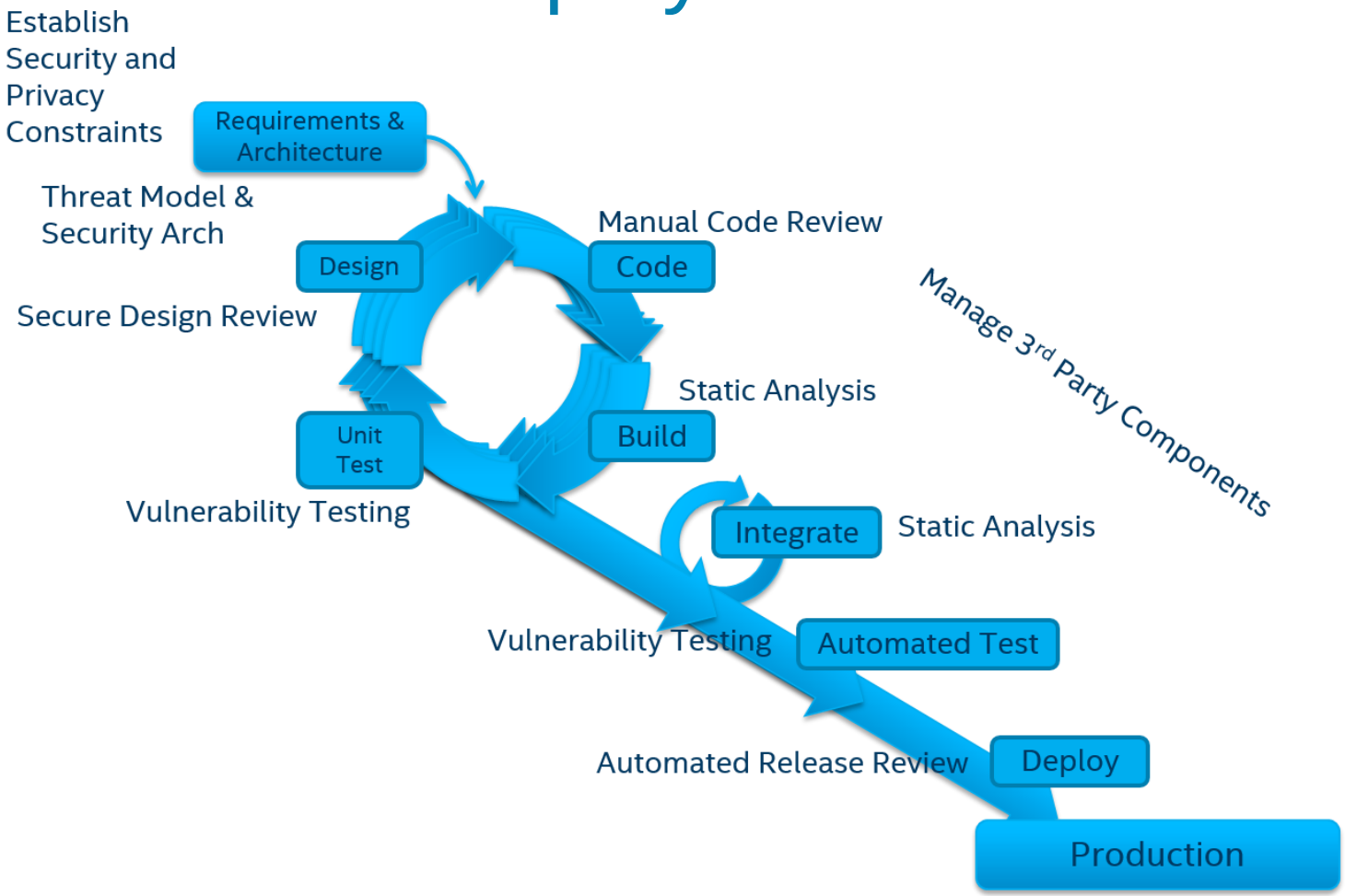
SSDF in Iterative Development



Secure Software Dev. Framework

Damilare D. Fagbemi

SSDF in Continuous Deployment



SSDF Solutions to SDL Pain Points

Solution 1: Defined SDL Roles and Responsibilities

SDL pain points addressed:

- Pain 1: Developers don't fully understand their SDLC
- Pain 3: SDL <-> Product Development mapping unclear

Secure Software Dev. Framework

Damilare D. Fagbemi

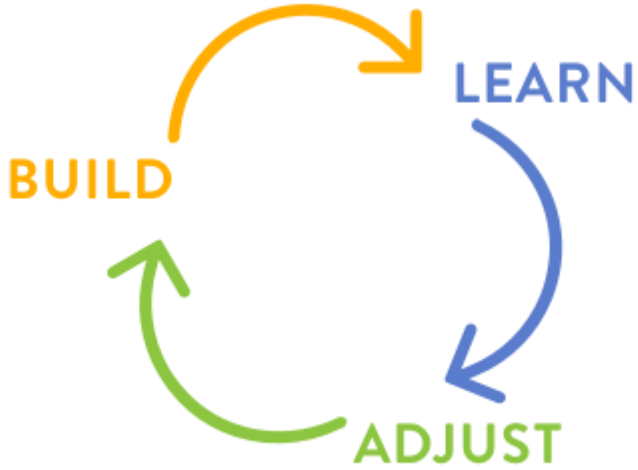
Solution 2: SSDF Activities are Strictly Aligned with Building and Shipping Code

<code wins arguments />

SDL pain points addressed:

- Pain 1: Developers don't fully understand their SDLC
- Pain 2: Security Engineers constrain SDL to their narrow/ specific SDLC understanding
- Pain 3: SDL <-> Product Development mapping unclear

Solution 3a: Re-visitation Guidance



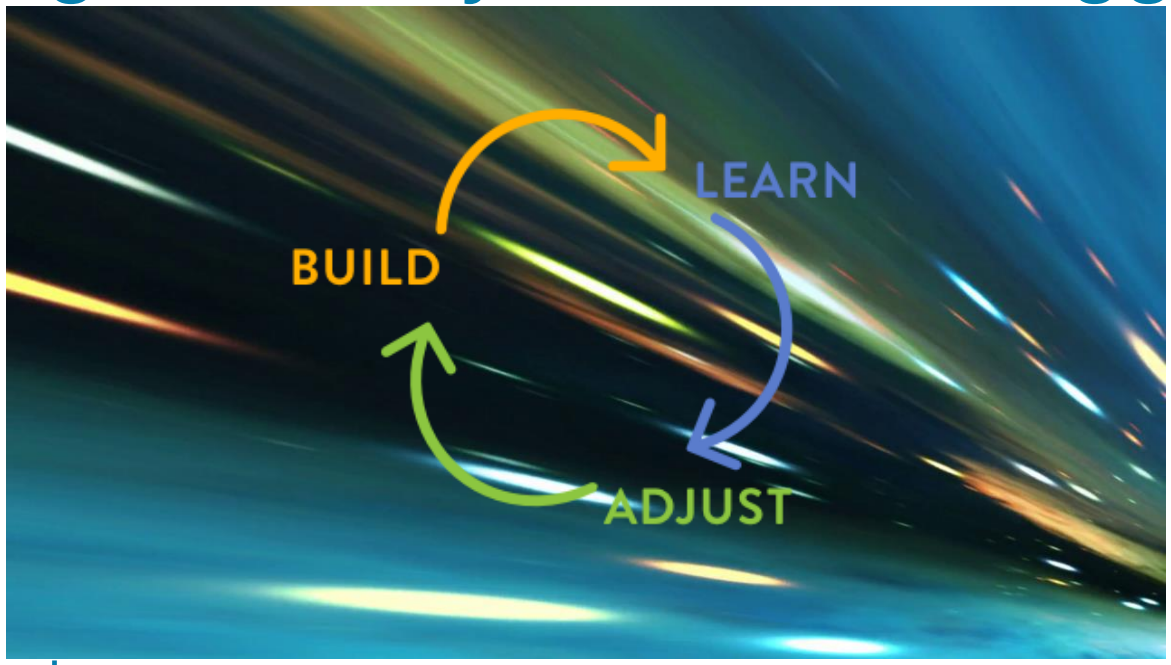
SDL pain points addressed:

- Pain 1: Developers don't fully understand their SDLC
- Pain 2: Security Engineers constrain SDL to their narrow/ specific SDLC understanding
- Pain 3: SDL <-> Product Development mapping unclear
- Pain 4: Inconsistency in SDL execution

Secure Software Dev. Framework

Damilare D. Fagbemi

Solution 3b: High Velocity Releases Trigger Re-visitation



SDL pain points addressed:

- Pain 1: Developers don't fully understand their SDLC
- Pain 2: Security Engineers constrain SDL to their narrow/ specific SDLC understanding
- Pain 3: SDL <-> Product Development mapping unclear
- Pain 4: Inconsistency in SDL execution

Recommendation X: Add Architecture & Design Derived Requirements to SDL

SDL pain points addressed:

- Pain 4: Ambiguous security requirements
- Pain 5: Inconsistency in SDL execution

Any Questions?

Thanks!

Where to Find Me:

- LinkedIn: Damilare D. Fagbemi
- Blog: edgeofus.com
- Twitter: @damilarefagbemi
- Email: damilarefagbemi@gmail.com

Backup

Examples of Re-visitation Guidance in SSDF

Activity: Threat Modeling & Security Architecture

- This activity should be revisited:
  - New/modified requirements or design changes are identified which have the potential to impact data flows, overall architectural structure or mitigations
  - Whenever a new threat is identified that was not previosly accounted for. (e.g., security validation identifying a new threat, PSIRT incident,...)
  - If the overall product architecture has undergone incremental minor changes over time but has not been reviewed holistically in a year or more

Examples of Re-visitation Guidance in SSDF

Activity: Secure Design Review

- This activity should be revisited:
  - This activity should be revisited when there are changes to the threat model or security architecture (including threats, mitigations, security/privacy objectives, priorities).

Examples of Re-visitation Guidance in SSDF

Activity: Manual Code Reviews

- This activity should be revisited:
  - Anytime changes are made to high risk code or its dependencies.
  - Any time existing code is used in a new or different way that elevates it to high risk.

Examples of Re-visitation Guidance in SSDF

Activity: Static Analysis

- This activity should be revisited:
  - If the code changed since the last scan, rescan prior to release build.

Examples of Re-visitation Guidance in SSDF

Activity: Vulnerability Testing

- This activity should be revisited:
  - Anytime changes are made to product design or implementation, re-assess and determine if additional action needed
  - If implementation of a security feature (e.g. crypto,) or API/library/dependencies has changed
  - If the software interface or API definition has been modified in anyway
  - If a new security related defect is discovered (PSIRT), re-assess and determine if additional action needed