Для чего нам использовать практики SDL

Михаил Мельников
Development Team Lead

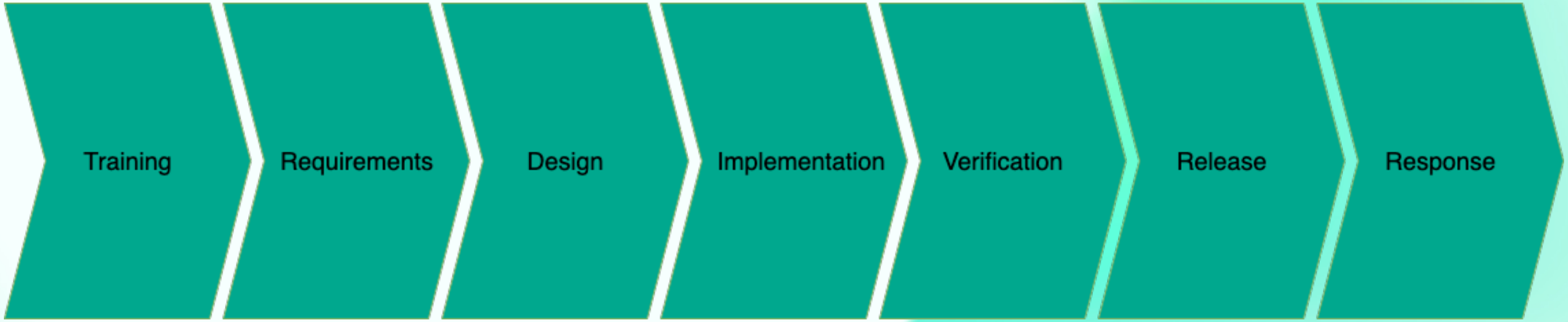
Security development lifecycle

Набор практик разработки программного обеспечения, помогающий разработчикам:

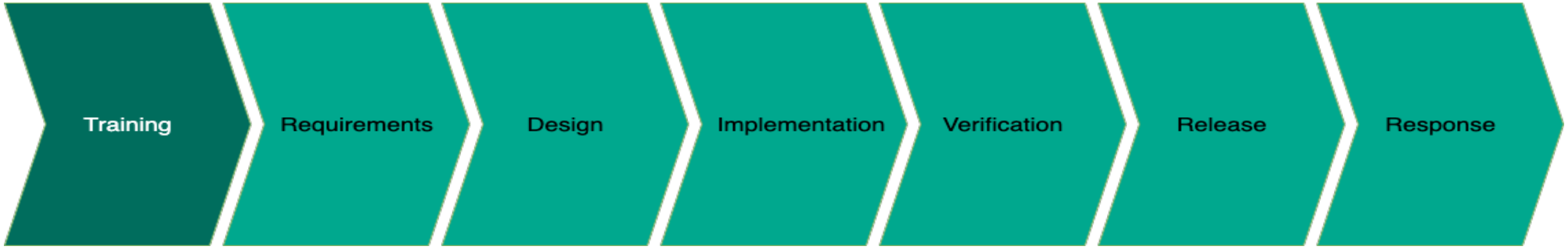
Создавать программное обеспечение, отвечающее требованиям безопасности

Снижать Затраты на разработку

Этапы цикла

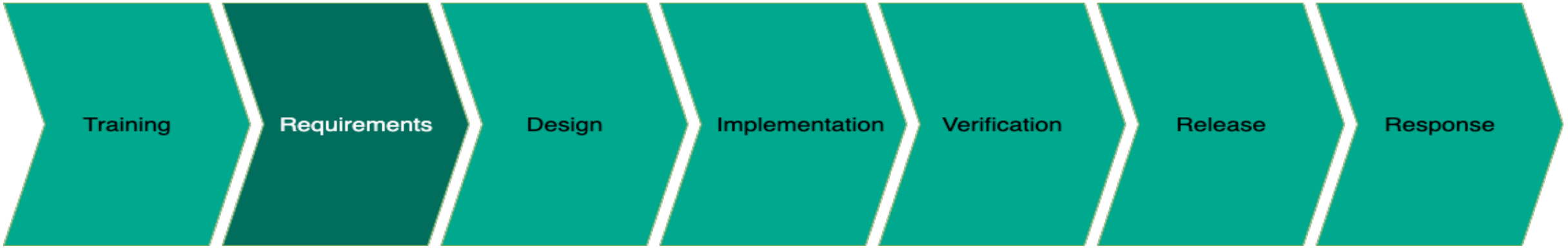

Training
Requirements
Design
Implementation
Verification
Release
Response

Этапы Цикла: Training



- Информирование
- Курсы
- Тестирование

Этапы Цикла: Requirements



- Выбор инструментария реализации
- Изначальные требования к безопасности
- Назначение областей ответственности

Этапы Цикла: Design

- Дизайн системы с учетом требований безопасности
- Моделирование угроз
- Снижение поверхности атак

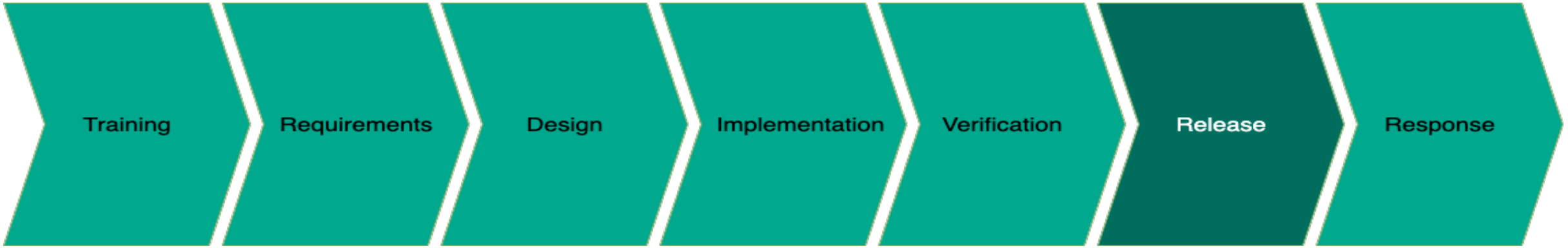
Этапы Цикла: Implementation

- Реализация продукта с помощью выбранного инструментария
- Постоянное поддержание актуального перечня используемых компонентов

Этапы Цикла: Verification

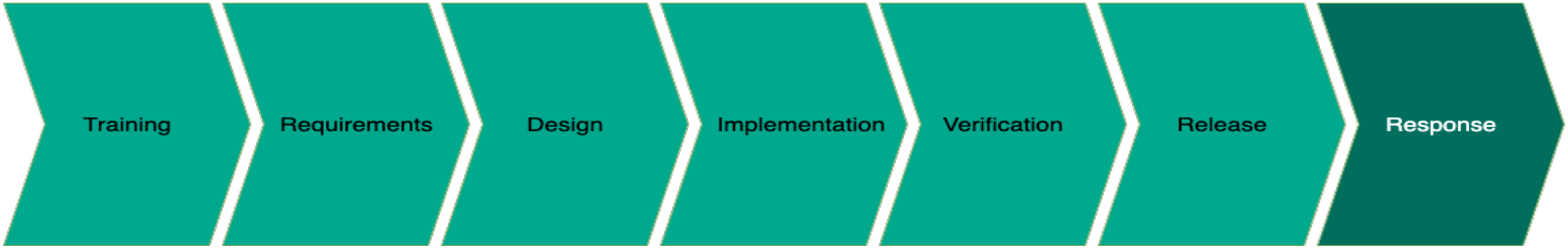
- Тестирование с учетом практик SDL
- Сравнение с заложенным на предыдущих этапах направлением
- Автоматизация

Этапы Цикла: Release



- Составление плана обработки инцидентов
- Final Security Review (FSR)

Этапы Цикла: Response



Обработка инцидентов безопасности в соответствии с планом

Этапы цикла

В которых может принимать участие разработчик

- Статический анализ кода
- Динамический анализ кода
- Анализ уязвимостей в стороннем коде
- Фаззинг тестирование
- Моделирование угроз
- Secure code review

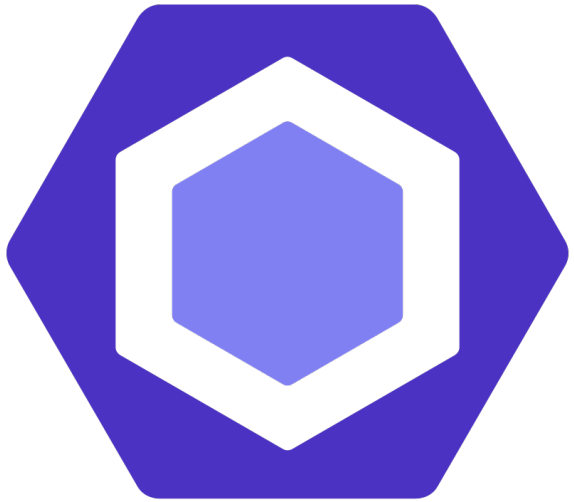
Security Champion

Отдельная роль в команде

Связующее звено между командами разработки и Security Team

Статический анализ кода

Инструмент: ESLint



Цель:

Автоматизированный анализ ошибок и уязвимостей в исходном коде программы

Мотивация:

- Использование общепринятого инструментария
- Правильно настроенный линтер может помочь в обнаружении уязвимостей (eslint-plugin-security)

Динамический анализ кода

Инструменты:

- Стандартный инструментарий профилирования NodeJS
- AppVerifier
- OpenSource библиотеки
- Внутренние утилиты

Цель:

Анализ ошибок и уязвимостей в момент выполнения программы

Мотивация:

- Освоение средств профилирования
- Углубление понимания работы приложения

Анализ уязвимостей в стороннем коде

Алгоритм:

- Сбор отчета по используемым библиотекам
- Анализ библиотек в реестре

Как у нас – ежесуточно обновляется реестр уязвимых библиотек, при отсутствии информации о библиотеке, она добавляется в реестр

Цель:

Отсутствие привнесенного уязвимого или вредоносного кода в проекте

Мотивация:

- Знание об используемых в проекте библиотеках и возможность отслеживать их динамику
- Отказ от легаси
- Целый мир CVE и их митигации

CVE*

cve.mitre.org

vuldb.com

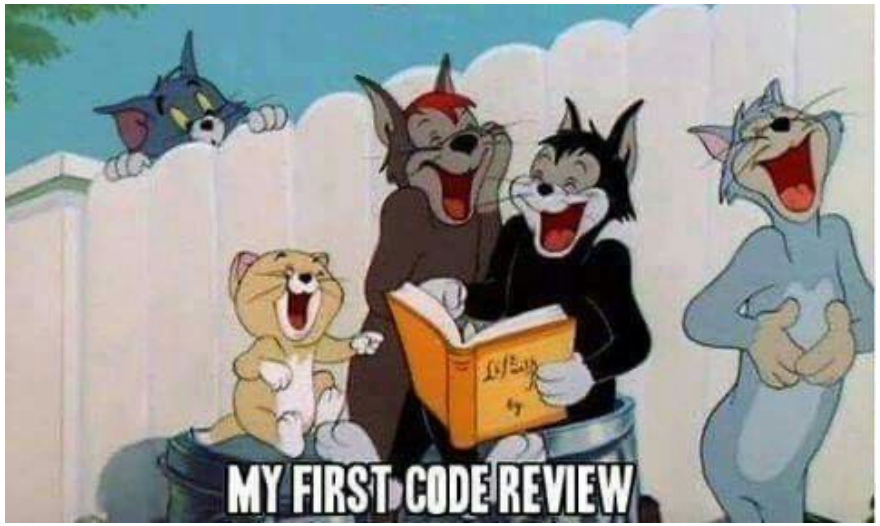
snyk.io

nvd.nist.gov

*Common Vulnerabilities and Exposures

Secure code review

Процесс анализа кода перед влитием его в целевые ветки



Мотивация:

- Пересмотр политик review
- Новые подходы к кодированию
- Ознакомление с кодовой базой

Fuzzing тестирование

Способ тестирования ПО, при котором на вход подаются заведомо неправильные или случайные данные

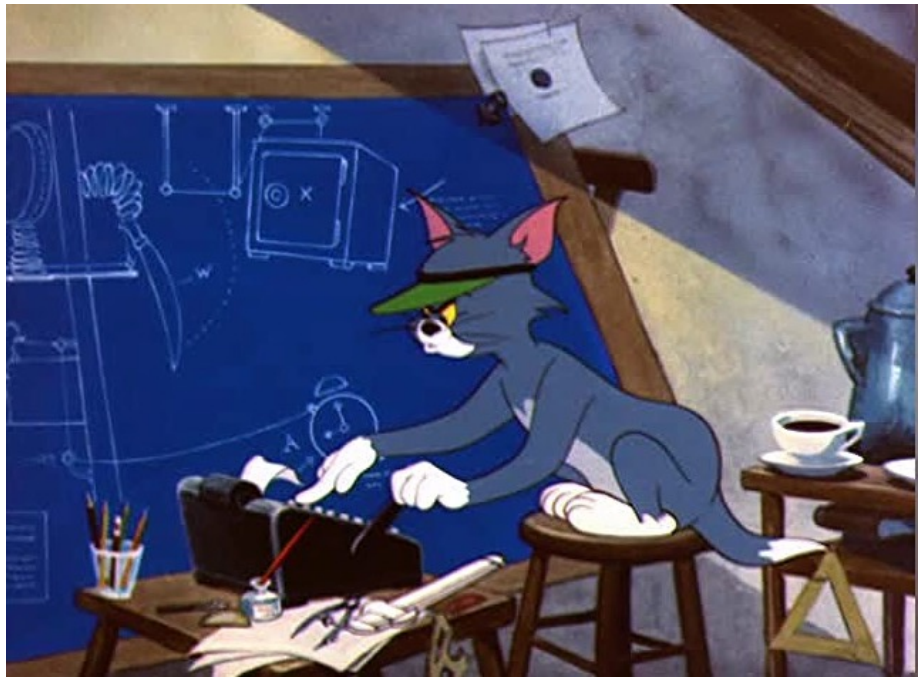
Инструменты:

- JSFuzz
- Jazzer

Мотивация:

- Умение фаззить
- Осознание того, что может быть использовано в качестве аргументов

Моделирование угроз



Цель:

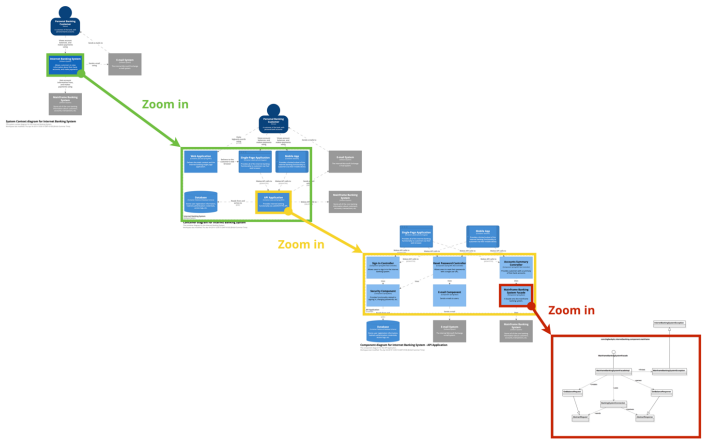
Проработка архитектуры системы покомпонентно, выявление типовых сценариев каждого блока схемы, применимость сценариев атаки.

Мотивация:

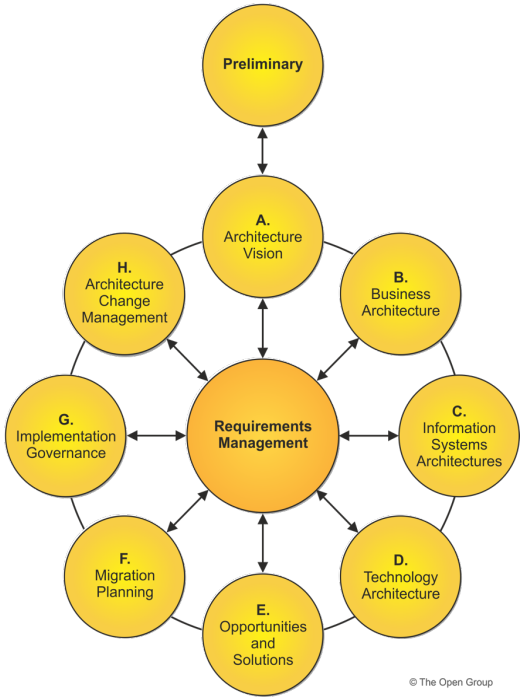
- Развитие своего навыка в архитектуре ПО
- Углубление понимания продуктовых сценариев

Инструментарий для отображения архитектуры приложения

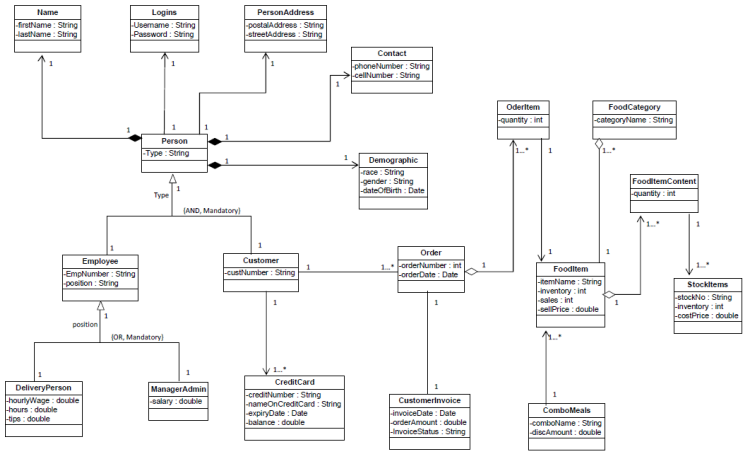
C4



TOGAF

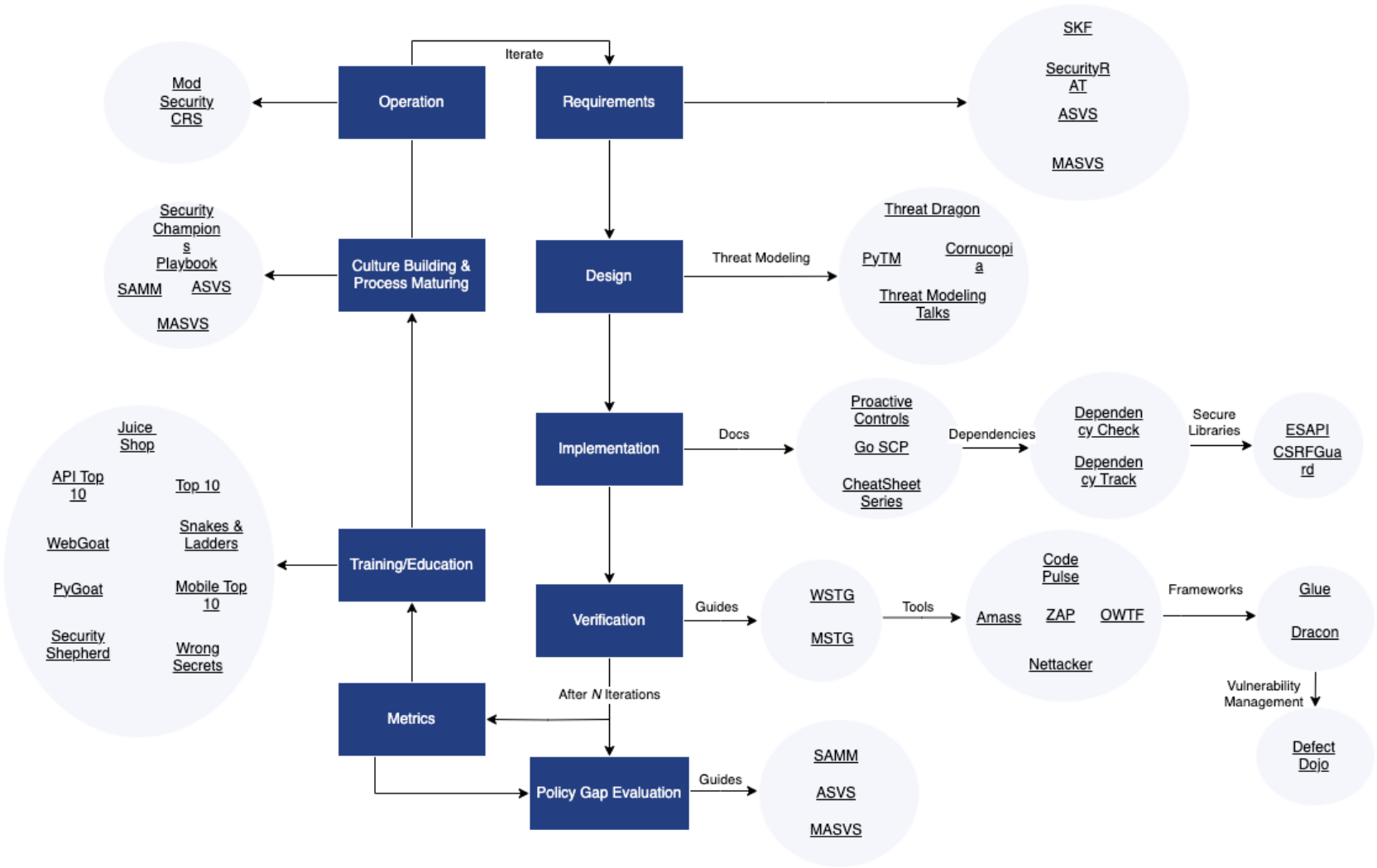


UML

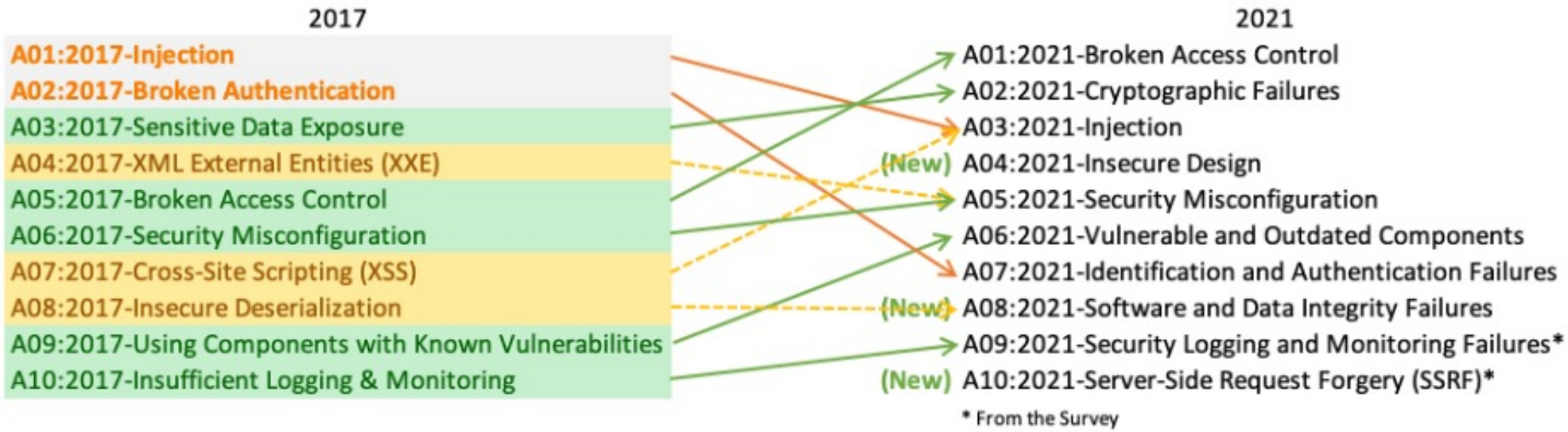


OWASP
Open Web Application Security Project

- API Security Project
- WrongSecrets
- etc



OWASP Top 10


2017
A01:2017-Injection
A02:2017-Broken Authentication
A03:2017-Sensitive Data Exposure
A04:2017-XML External Entities (XXE)
A05:2017-Broken Access Control
A06:2017-Security Misconfiguration
A07:2017-Cross-Site Scripting (XSS)
A08:2017-Insecure Deserialization
A09:2017-Using Components with Known Vulnerabilities
A10:2017-Insufficient Logging & Monitoring
2021
A01:2021-Broken Access Control
A02:2021-Cryptographic Failures
A03:2021-Injection
(New) A04:2021-Insecure Design
A05:2021-Security Misconfiguration
A06:2021-Vulnerable and Outdated Components
A07:2021-Identification and Authentication Failures
(New) A08:2021-Software and Data Integrity Failures
A09:2021-Security Logging and Monitoring Failures*
(New) A10:2021-Server-Side Request Forgery (SSRF)*
* From the Survey

Итог.
SDL для разработчика.

- T-Shape
- Мотивирует развивать продукт
- Облегчает maintenance
- Дает более глубокое понимание продукта

Resources

- https://www.microsoft.com/en-us/securityengineering/sdl
- https://owasp.org/
- https://snyk.io/
- https://cve.mitre.org/
- https://c4model.com/

Спасибо за внимание

Михаил Мельников Development Team Lead @Foxtr0tUniform

kaspersky