

# Security Testing For Dev and Tester



# Security Testing

\



# Software Testing

\



# Testing ?

When to test ?

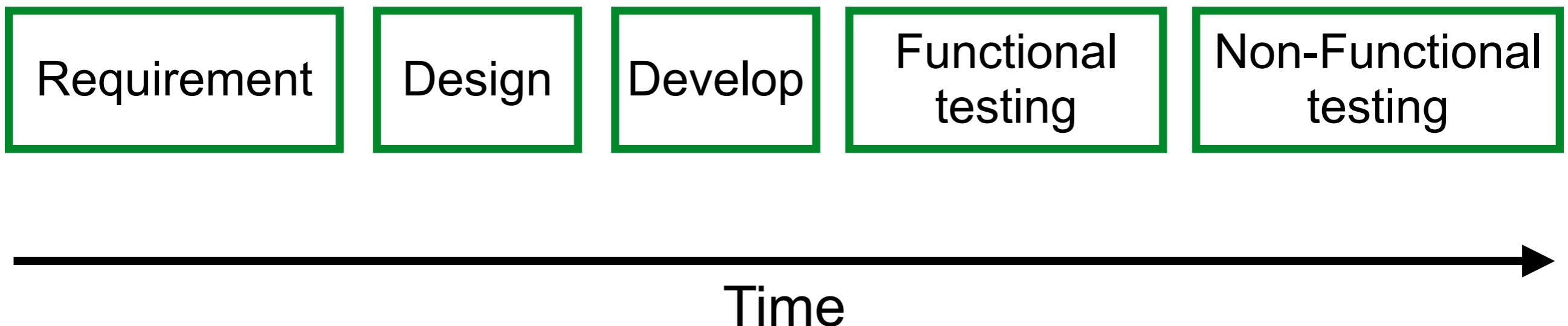
When to secure test ?



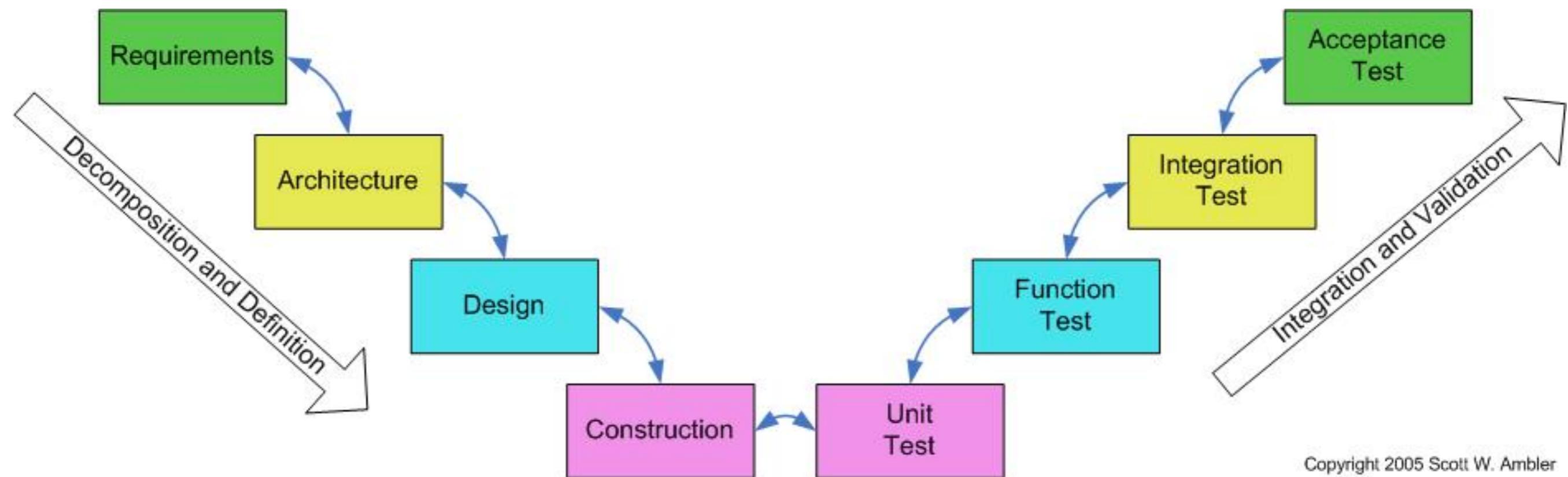
# Testing ?

When to test ?

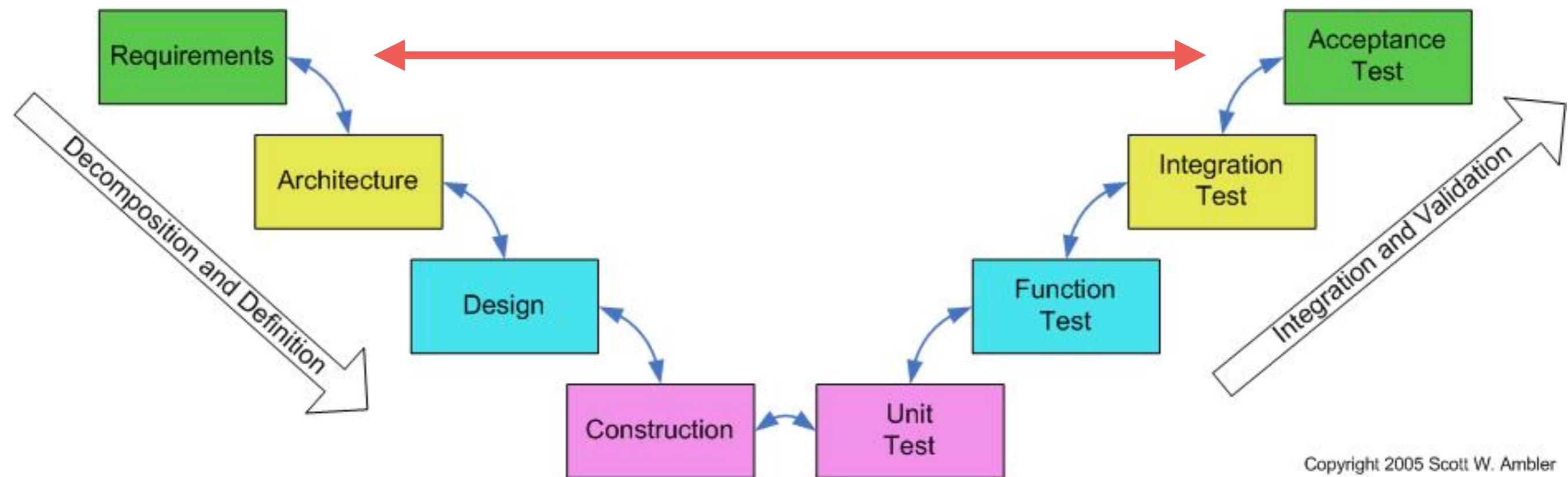
When to secure test ?



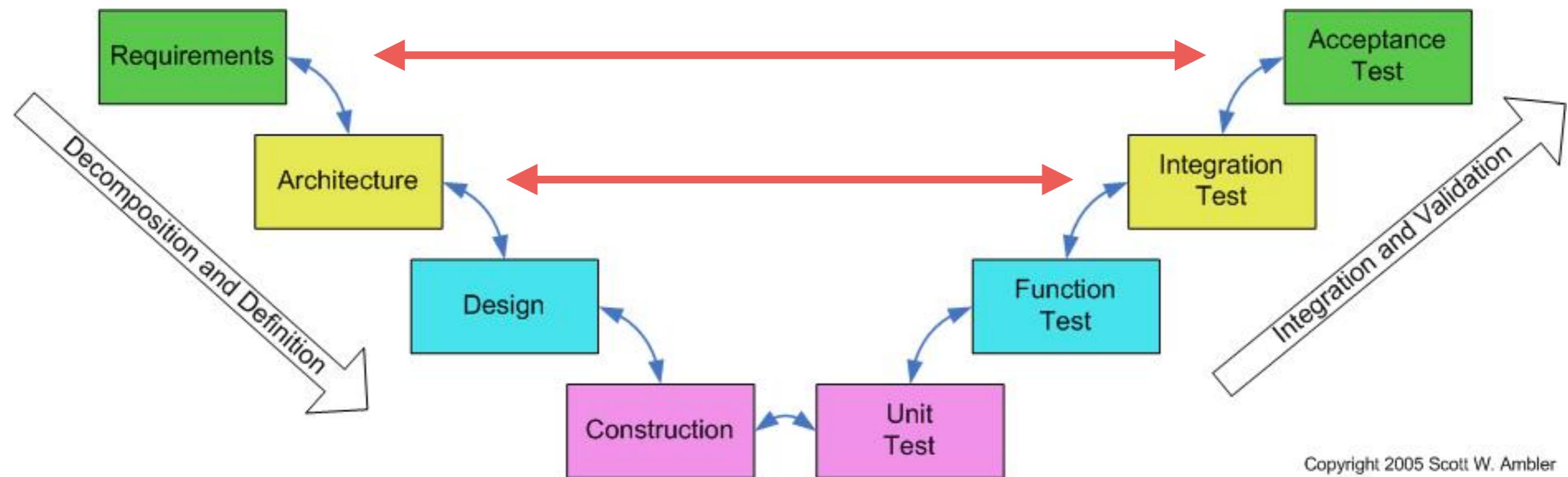
# V Model



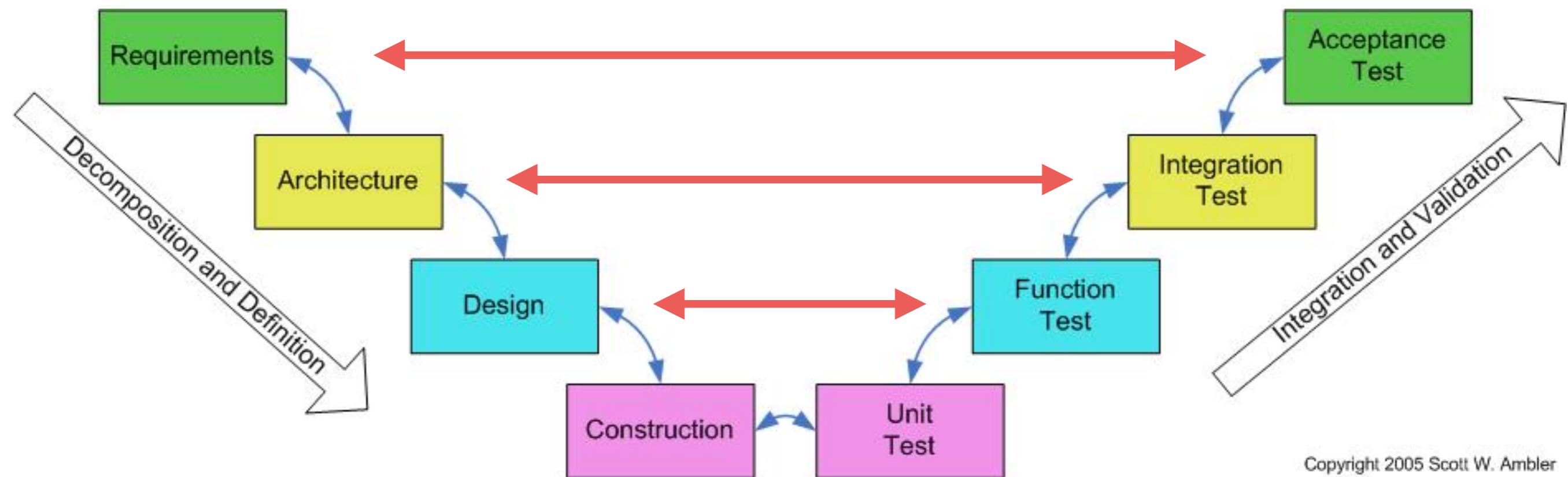
# V Model



# V Model



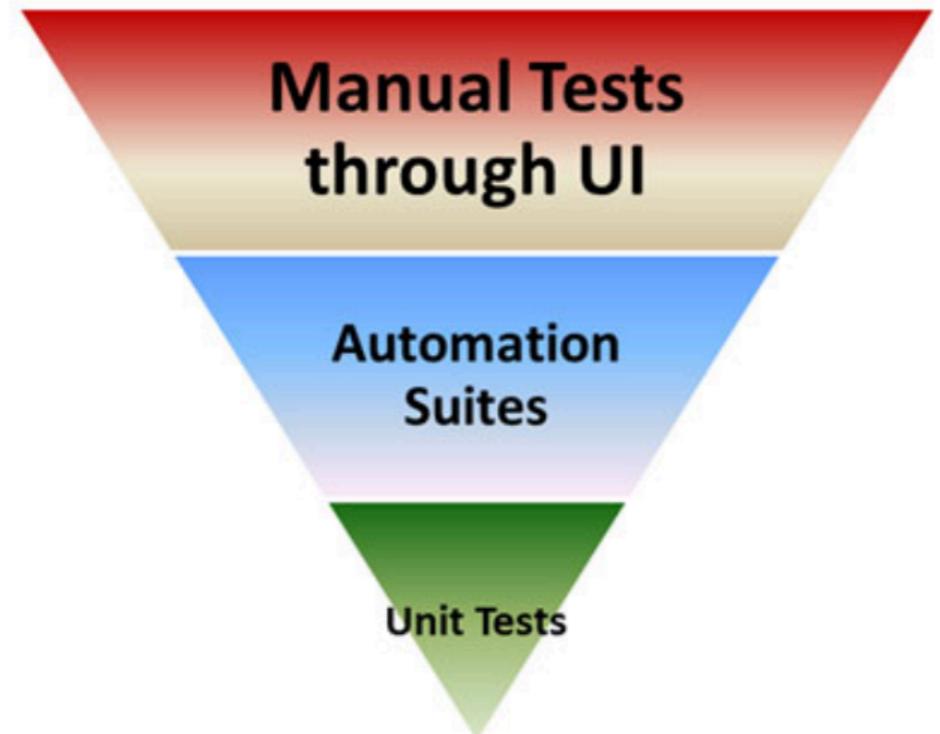
# V Model



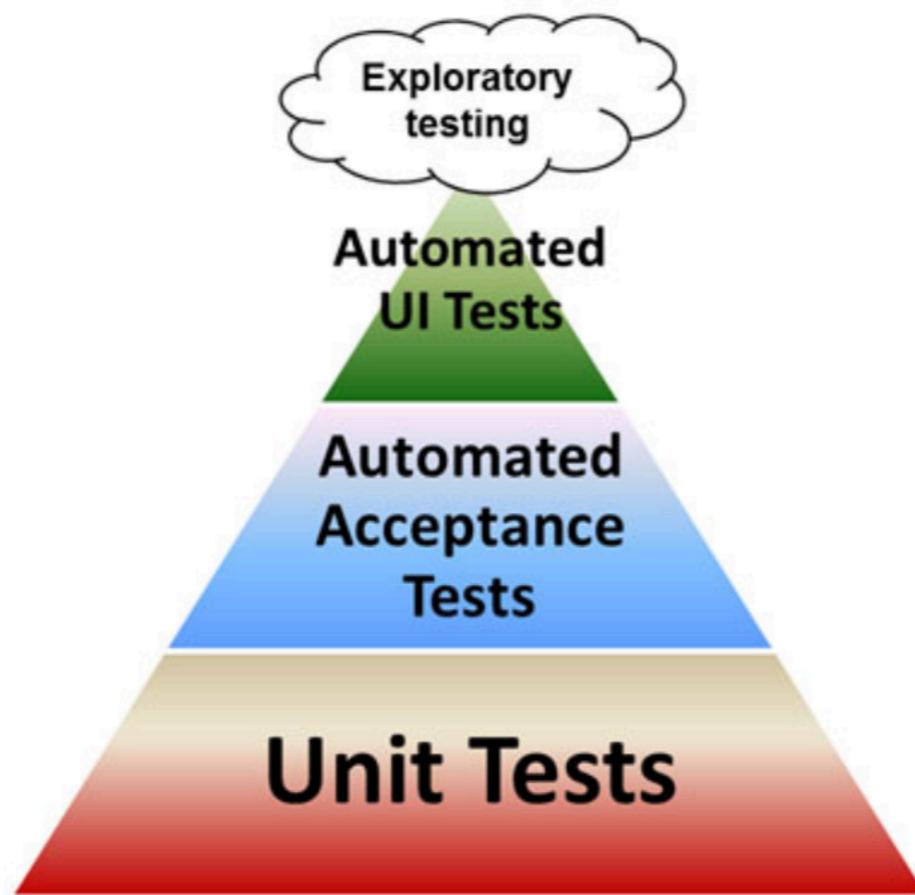
# Prevent vs Find

\





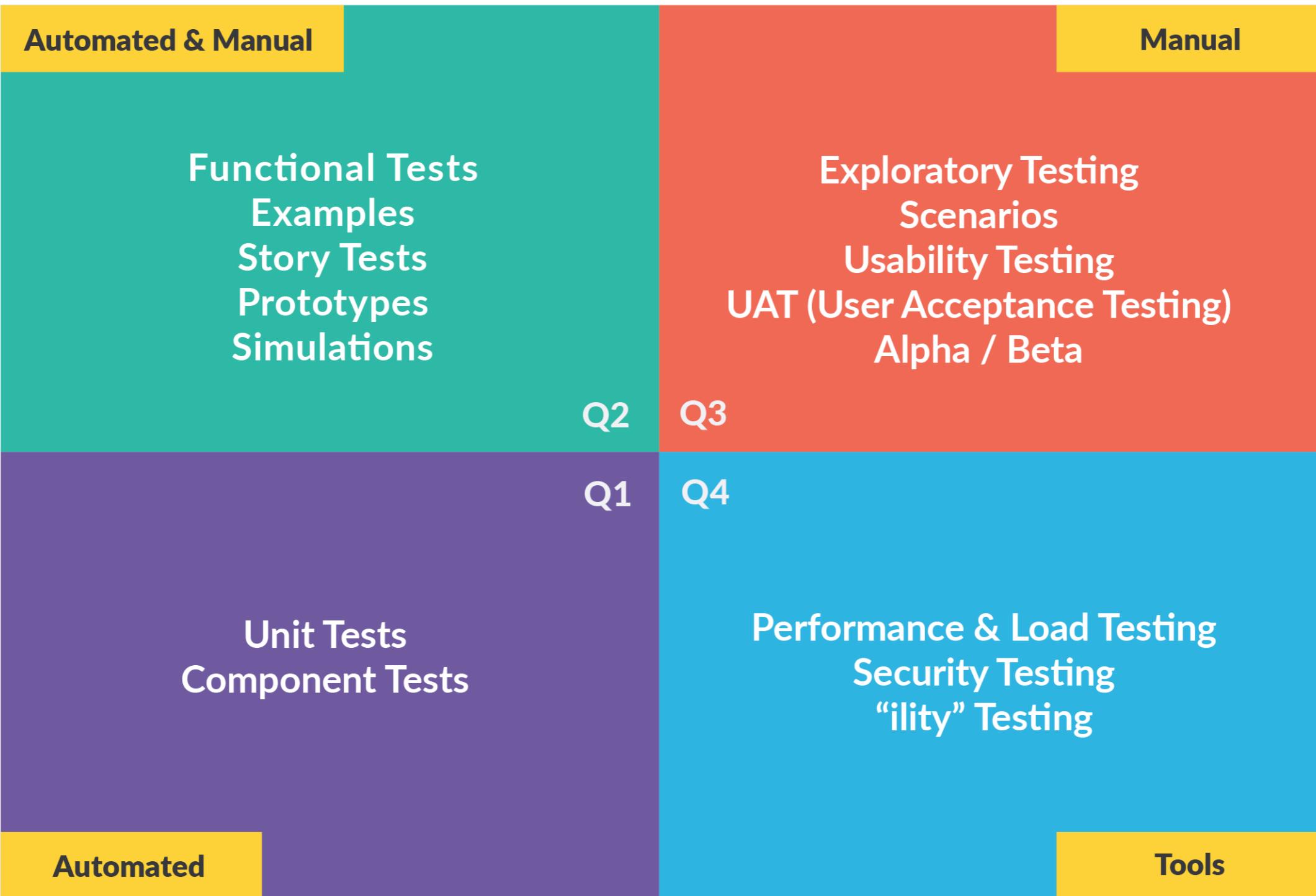
**Traditional**  
(find bugs)



**Agile**  
(prevent bugs)



## SUPPORTING THE TEAM



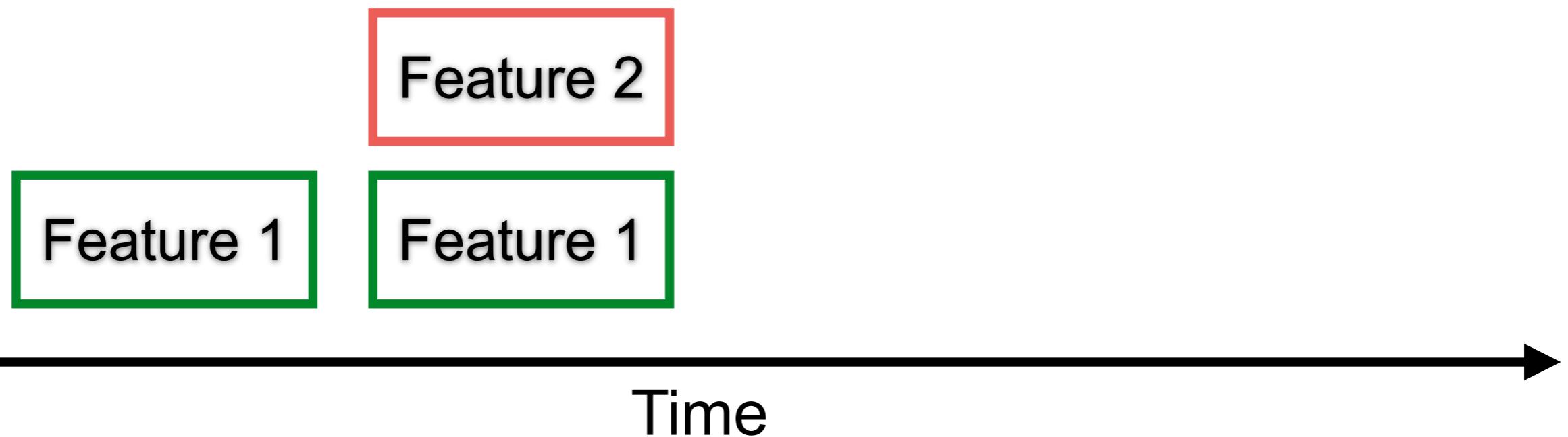
# Iterative and incremental process

Done = coded and tested



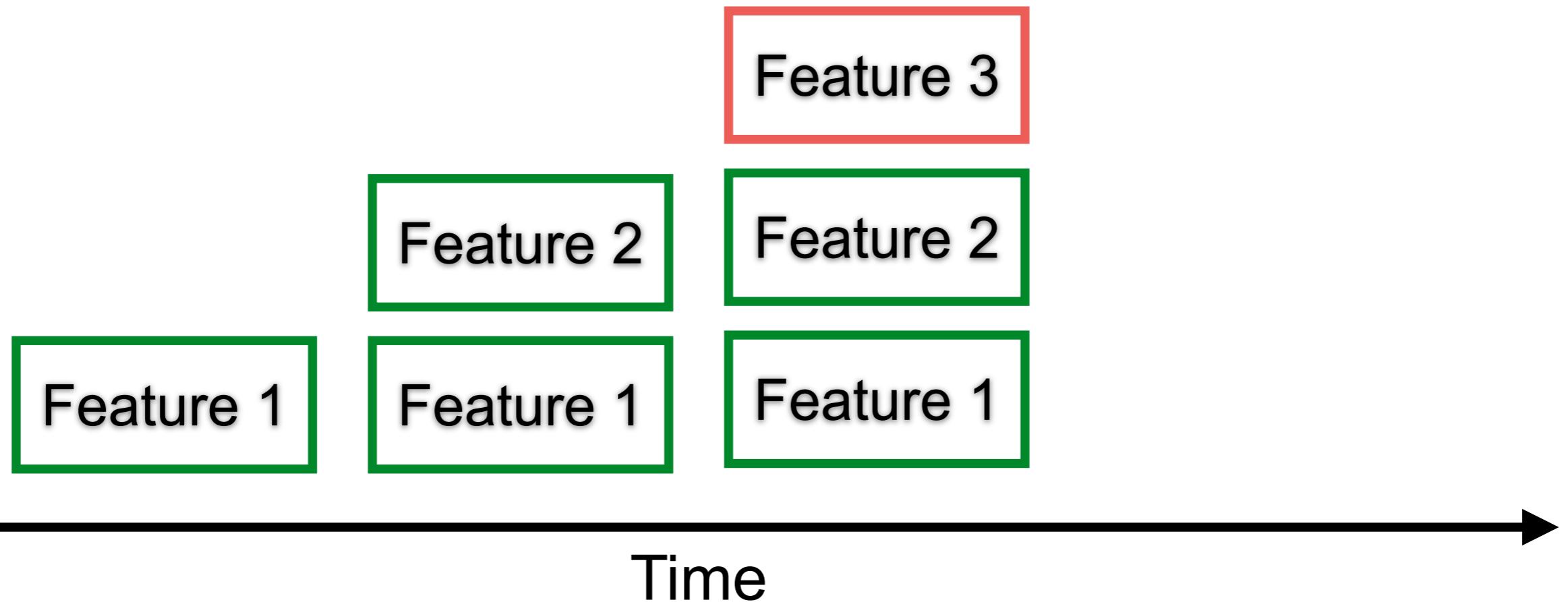
# Iterative and incremental process

Done = coded and tested



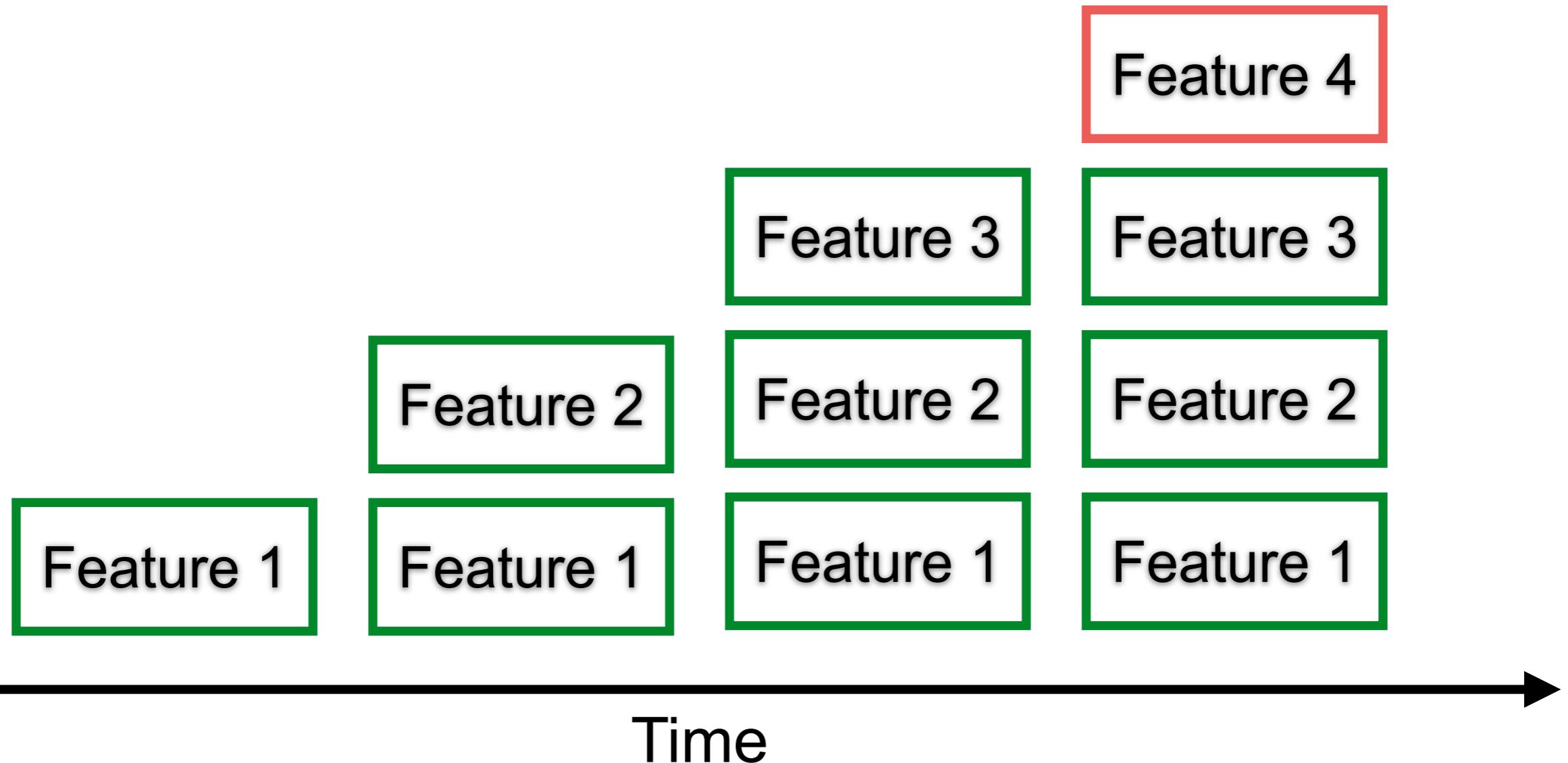
# Iterative and incremental process

Done = coded and tested



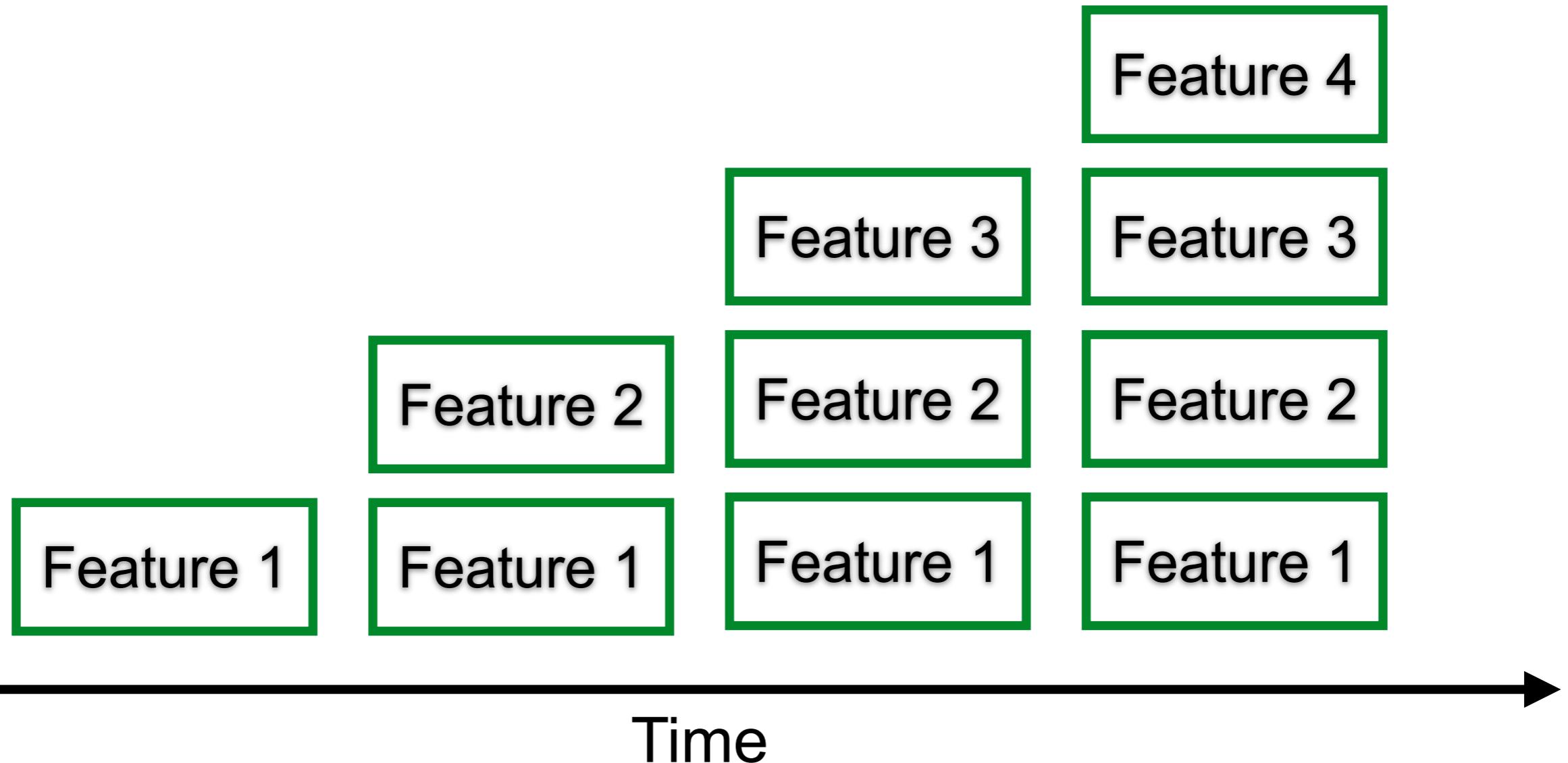
# Iterative and incremental process

Done = coded and tested

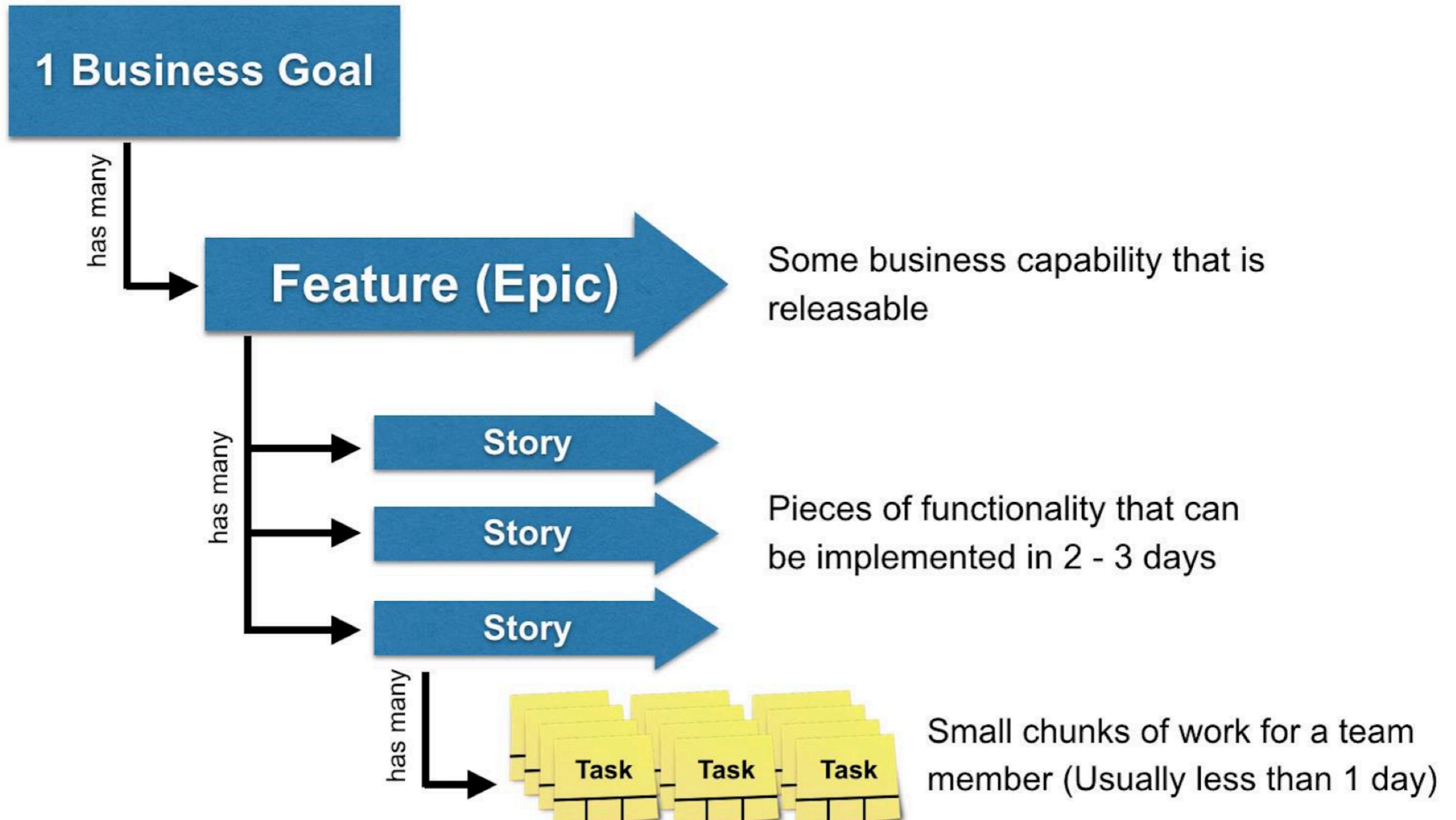


# Iterative and incremental process

Done = coded and tested

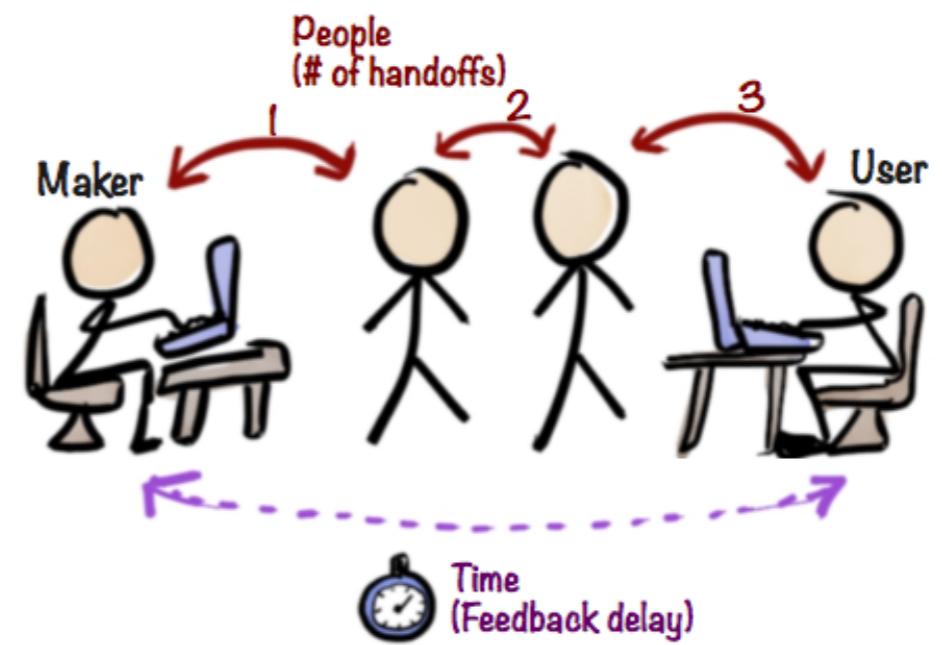
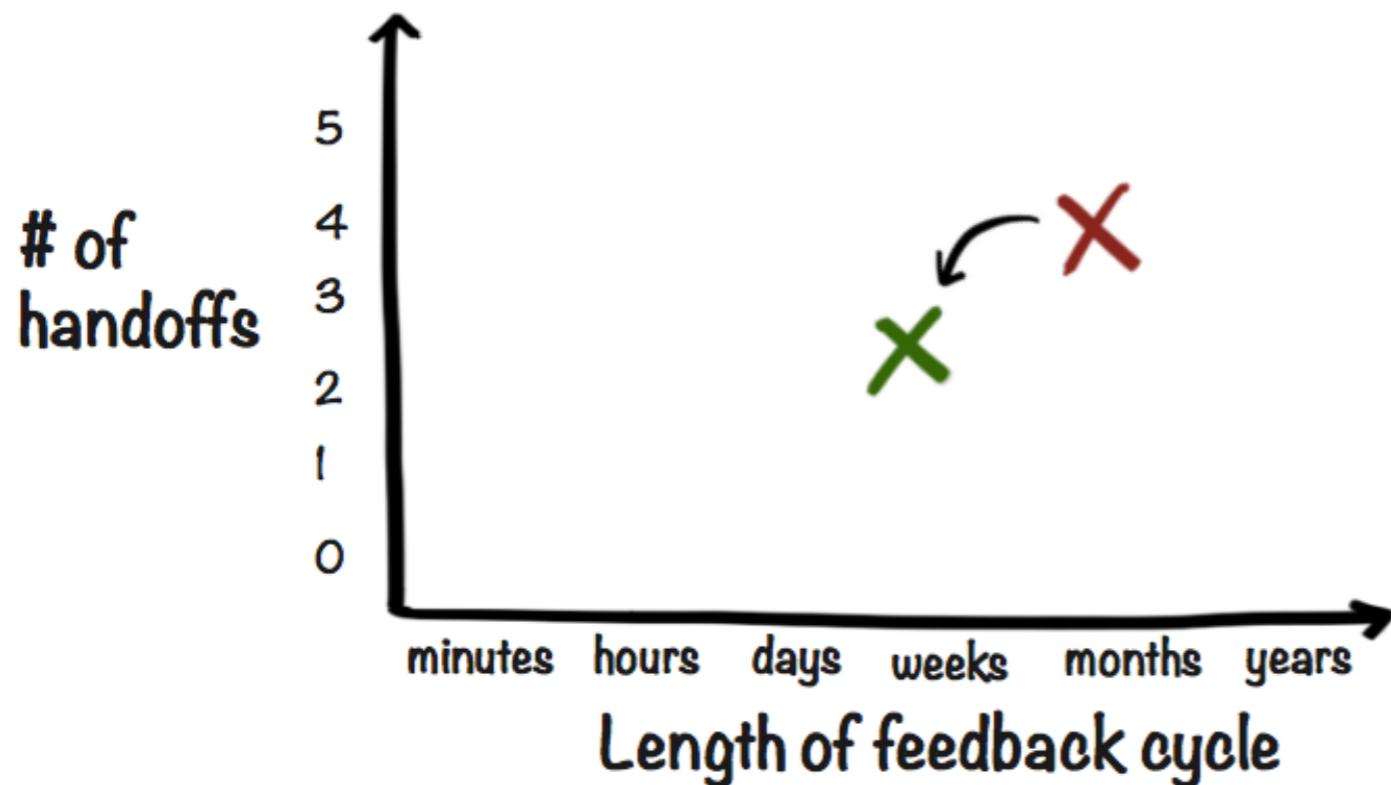


# Start with business goal to tasks

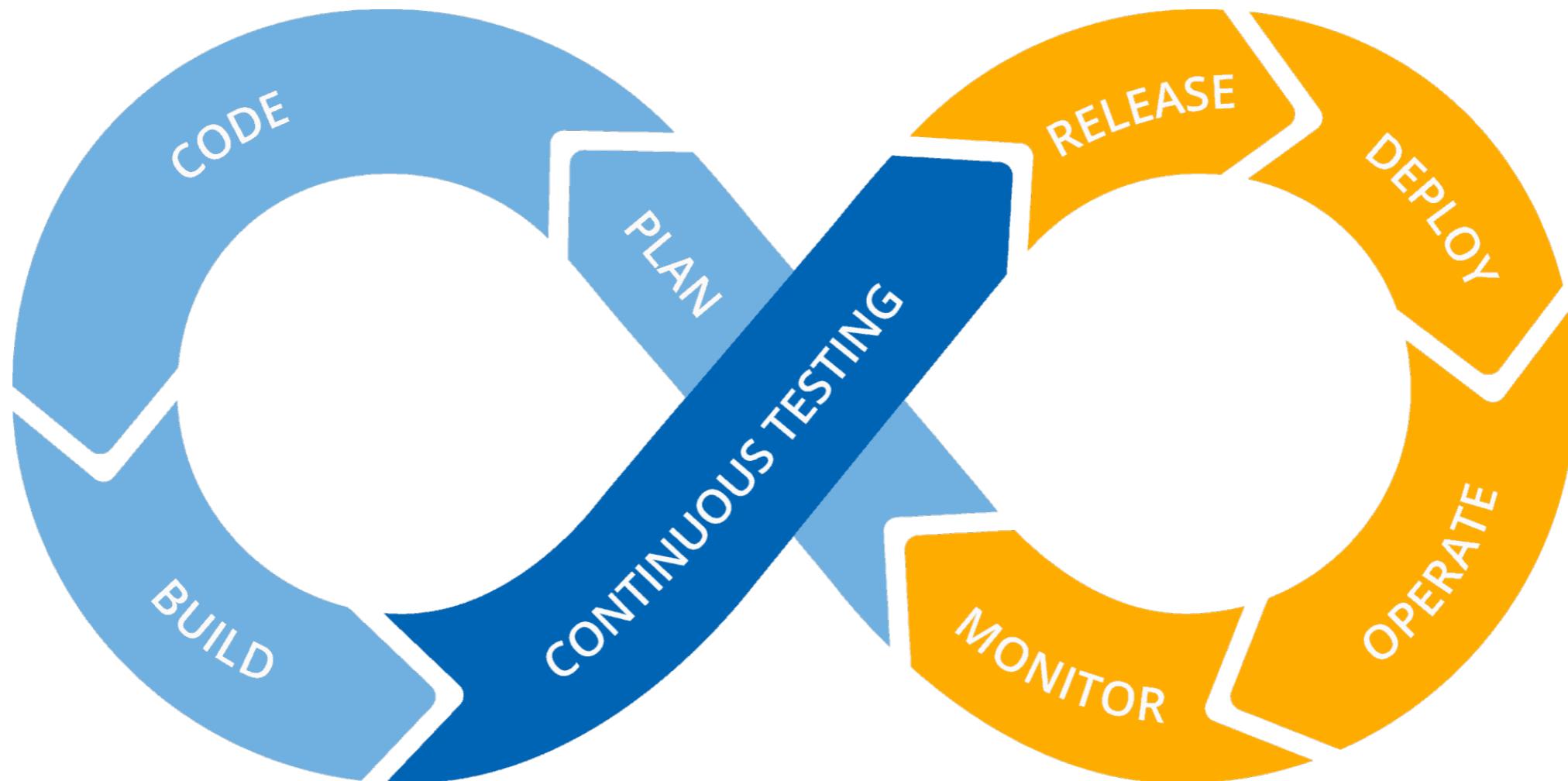


# Need Fast Feedback Loop

Shorten the feedback loop



# Continuous Testing

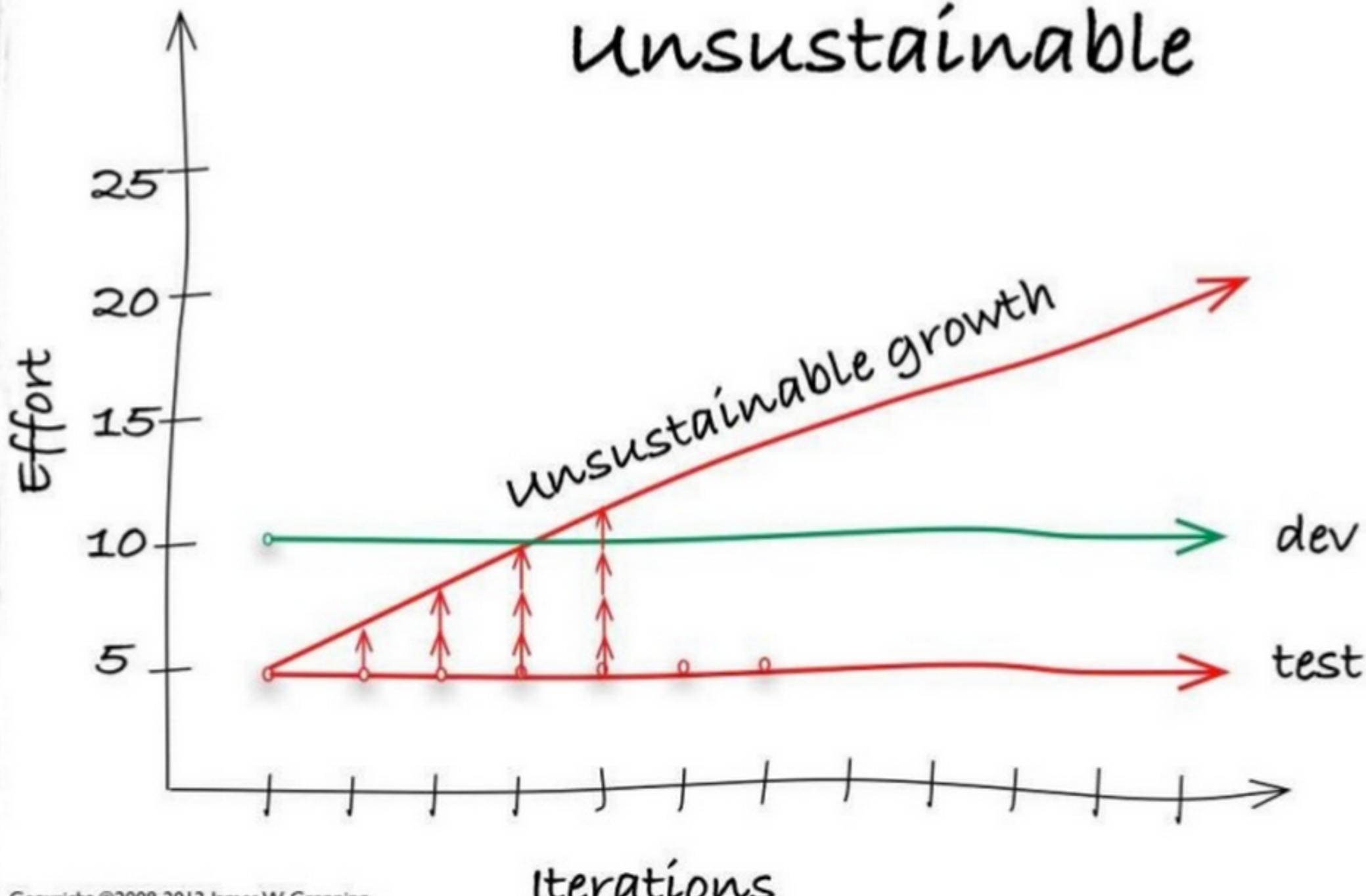


# Manual vs Automated testing

\



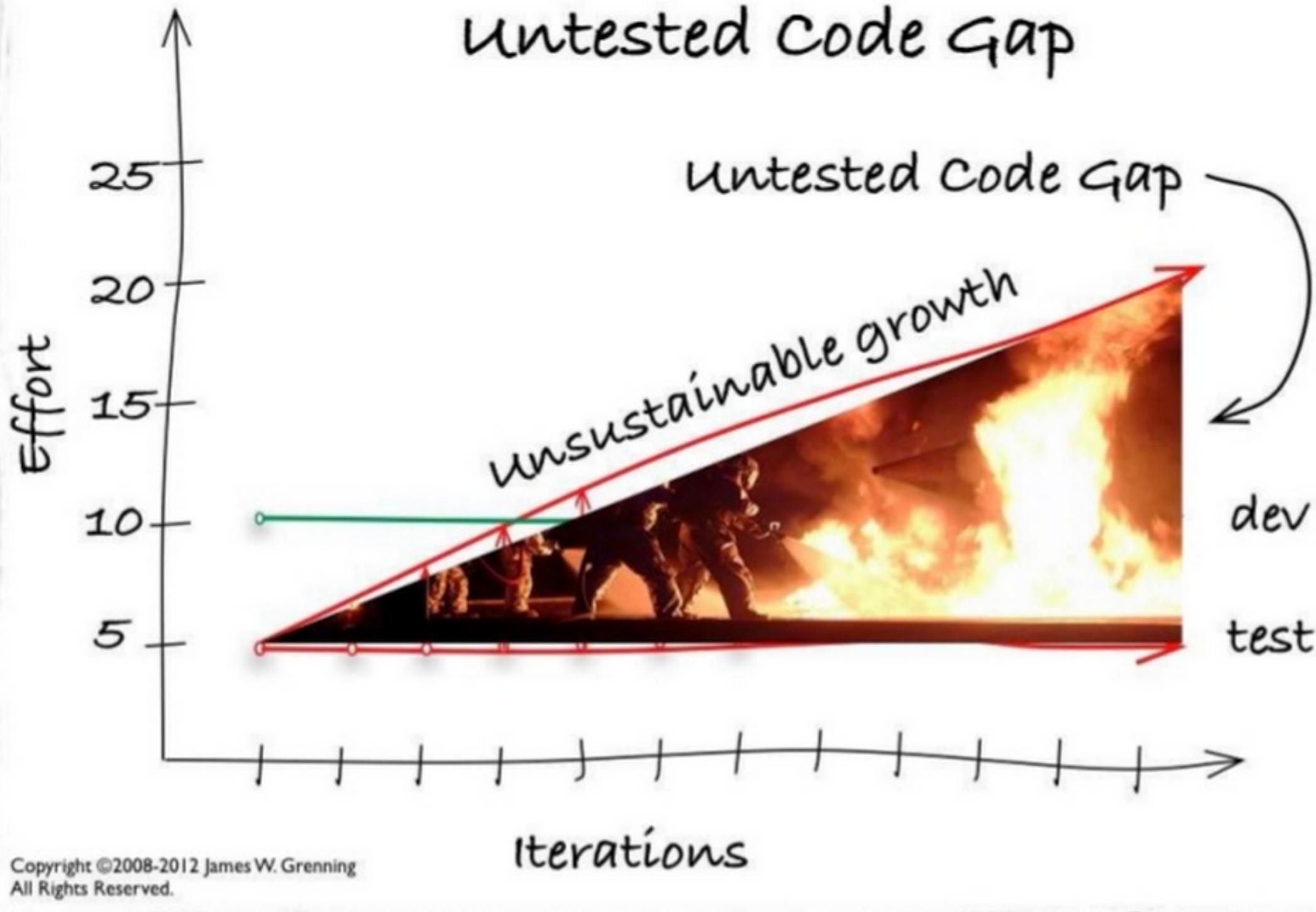
# Manual Test is unsustainable



Copyright ©2008-2012 James W. Grenning  
All Rights Reserved.



# Risk Accumulates in the Untested Code Gap



# Why we need automation ?

Manual checking take too long

Manual checks are error prone

Free people to do their best work (testing)

Living document

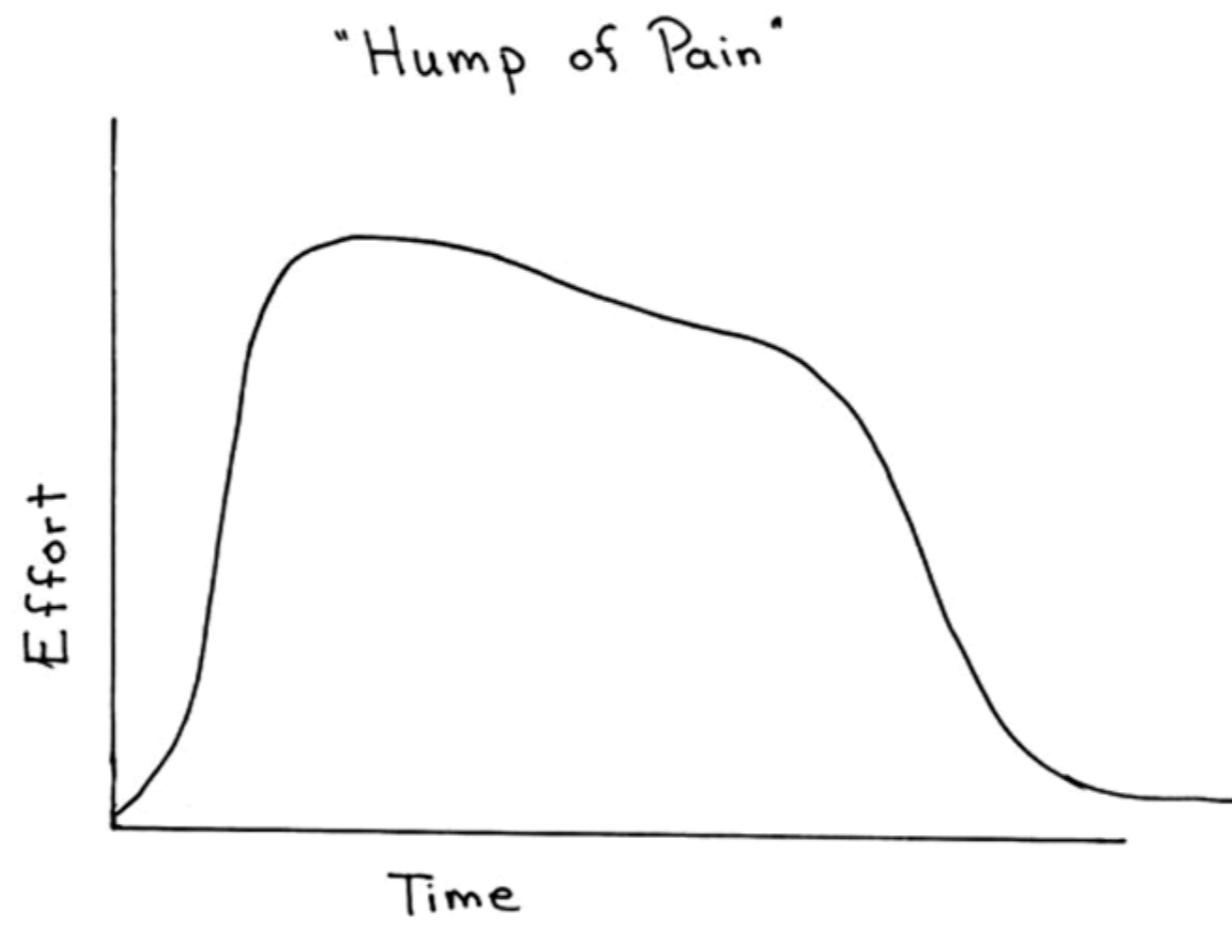
Repeatable and save time

\



# Initial investment !!

Hump of pain



\

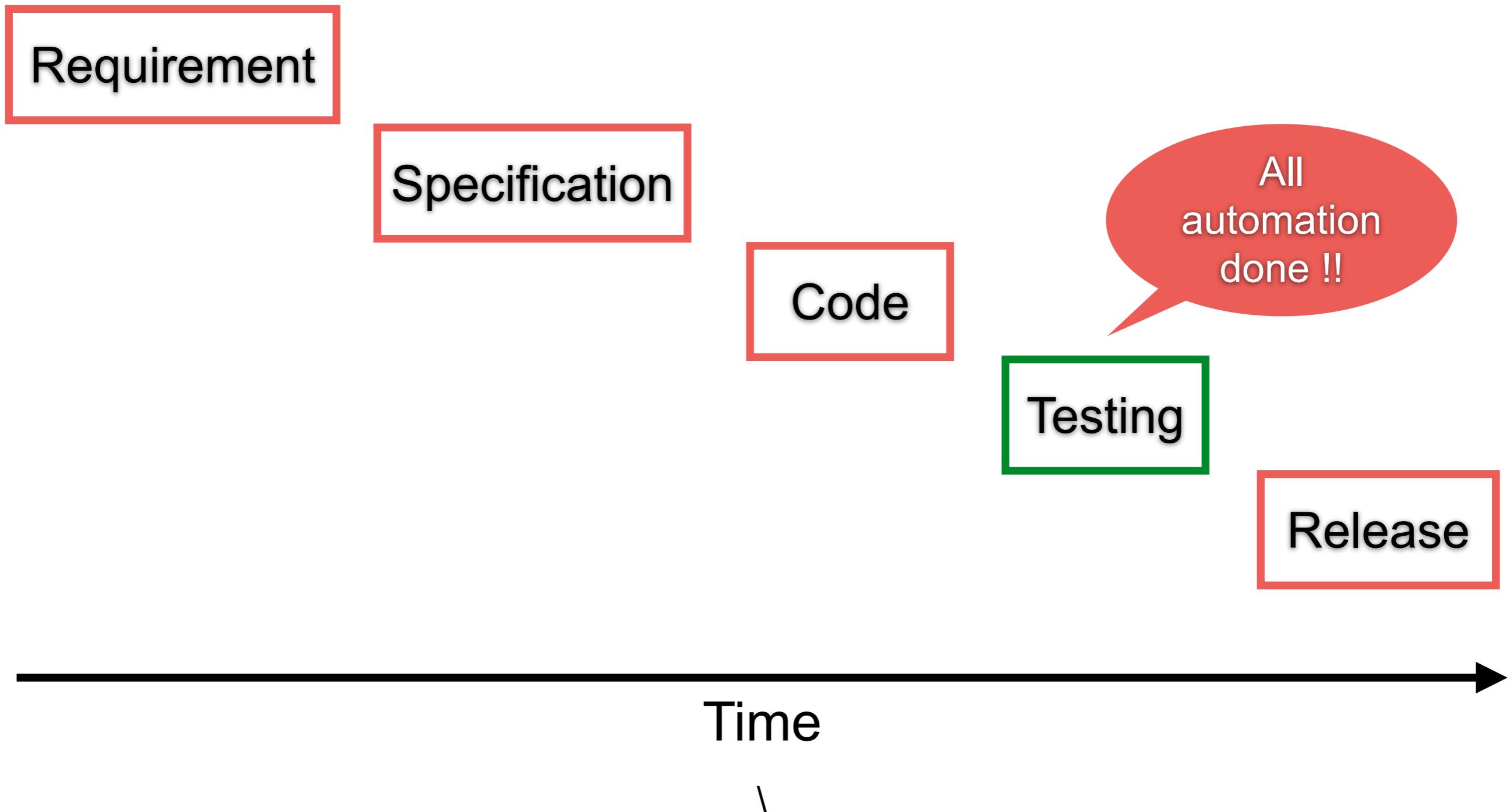


# Automation test strategy

\

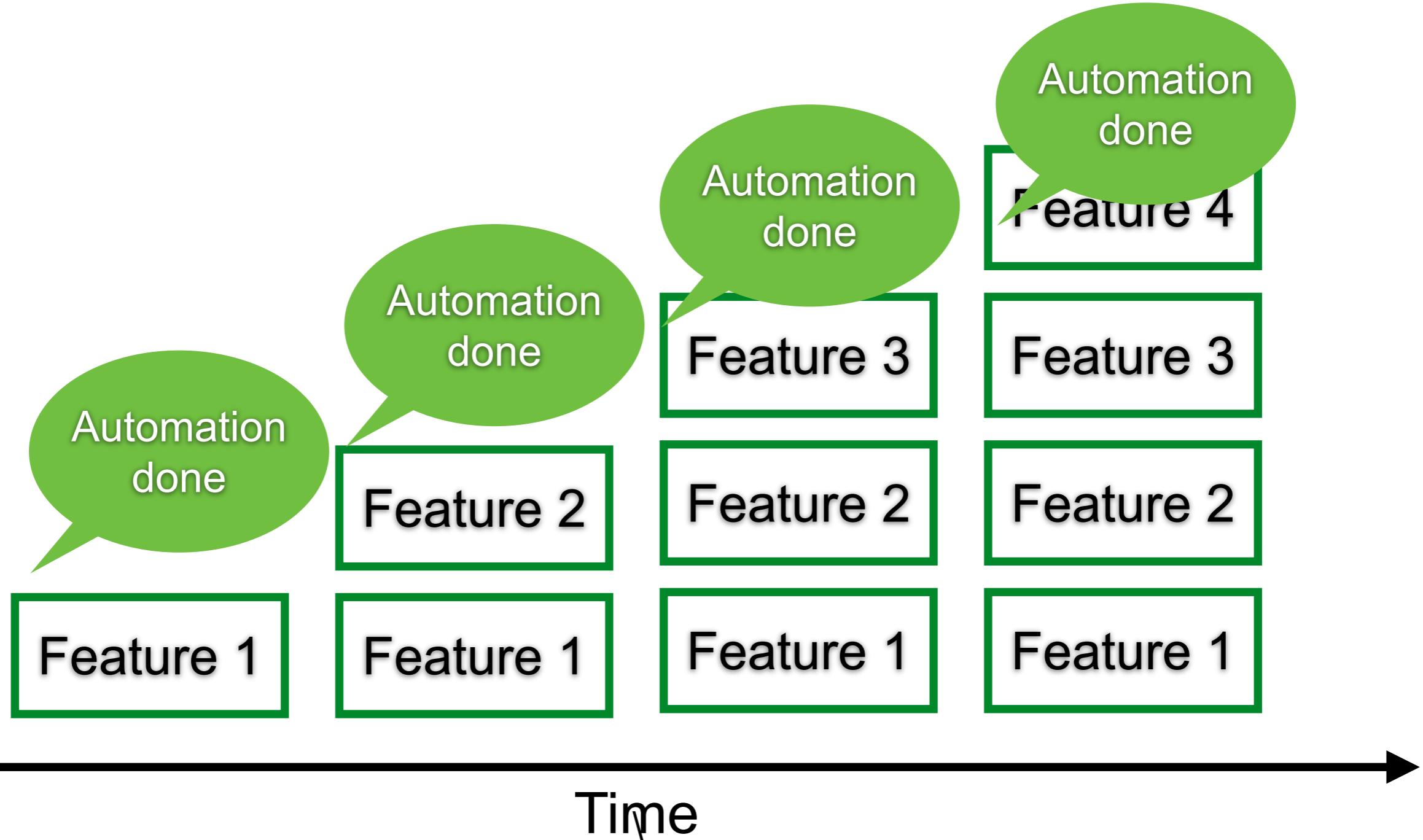


# Sequential process

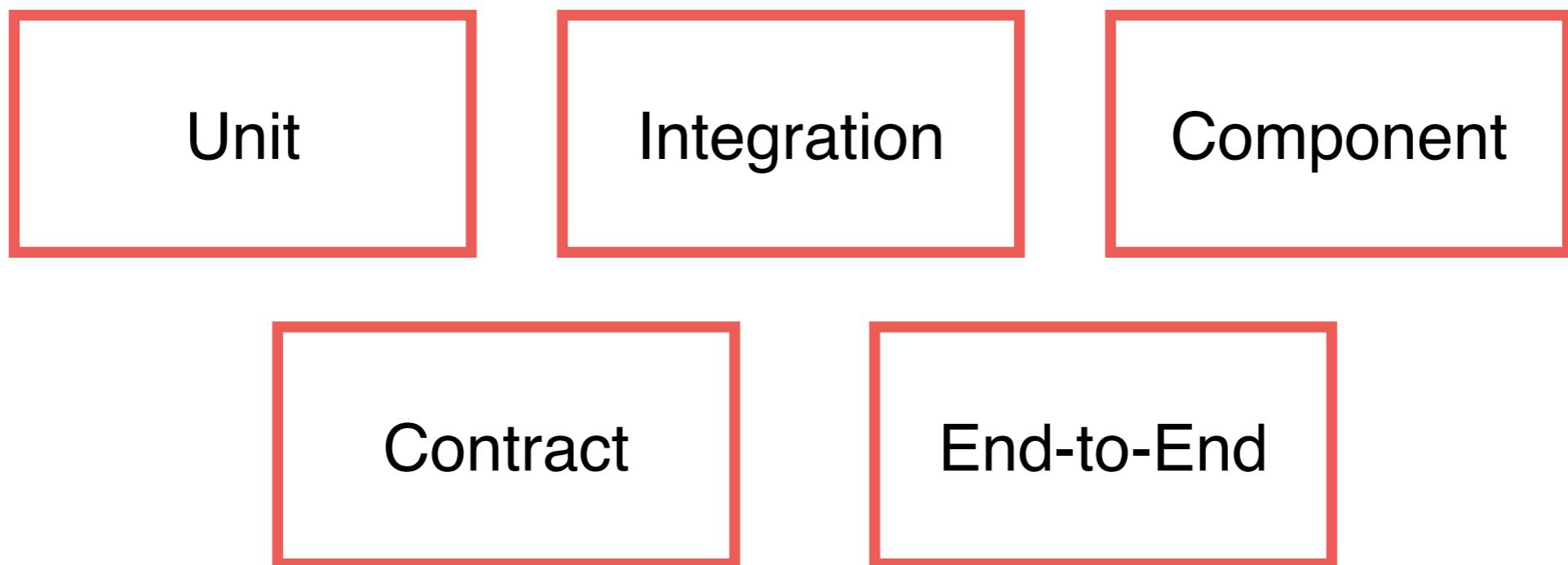


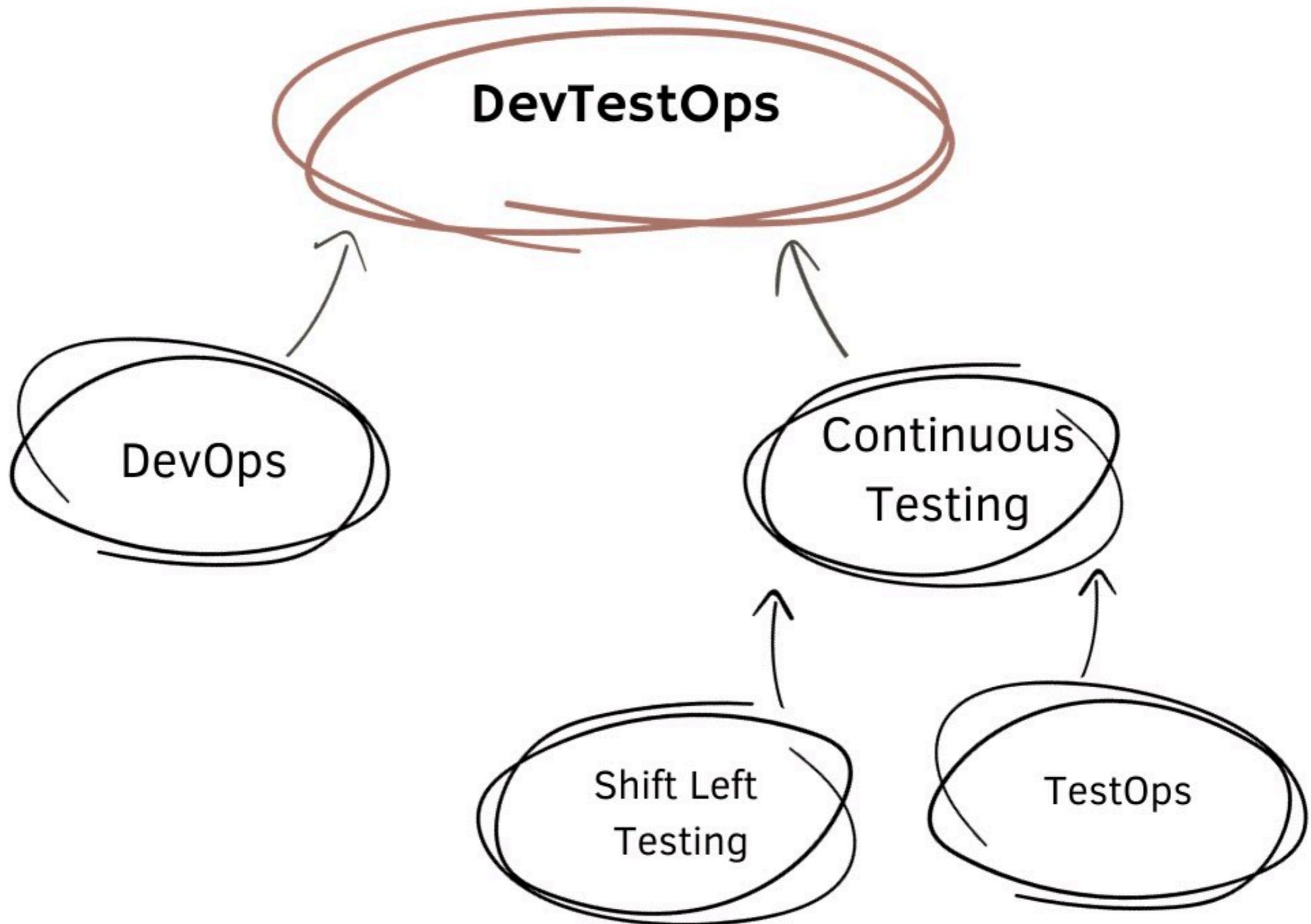
# Iterative and incremental process

Done = coded and tested



# Testing Types





# Security Testing

\



# Security Testing ?

Identify all possible weakness of system  
Impacts to system

Loss  
information

Loss revenue

Loss your  
business



# Types of Security Testing

Vulnerability  
Scanning

Security  
scanning

Penetration  
testing

Security  
auditing

Posture  
assignment

Ethical  
hacking



# SDLC Process



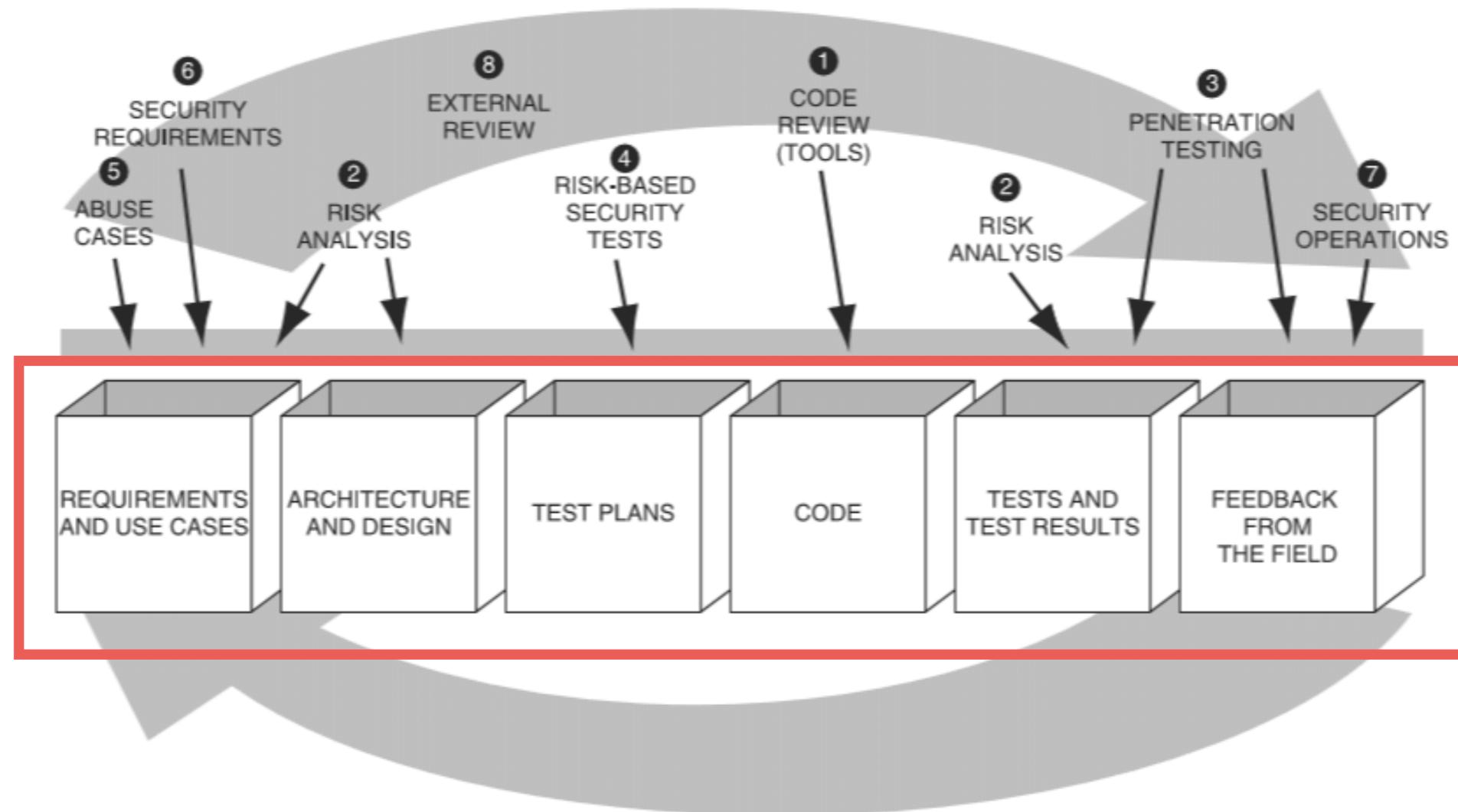
# Secure SDLC Process



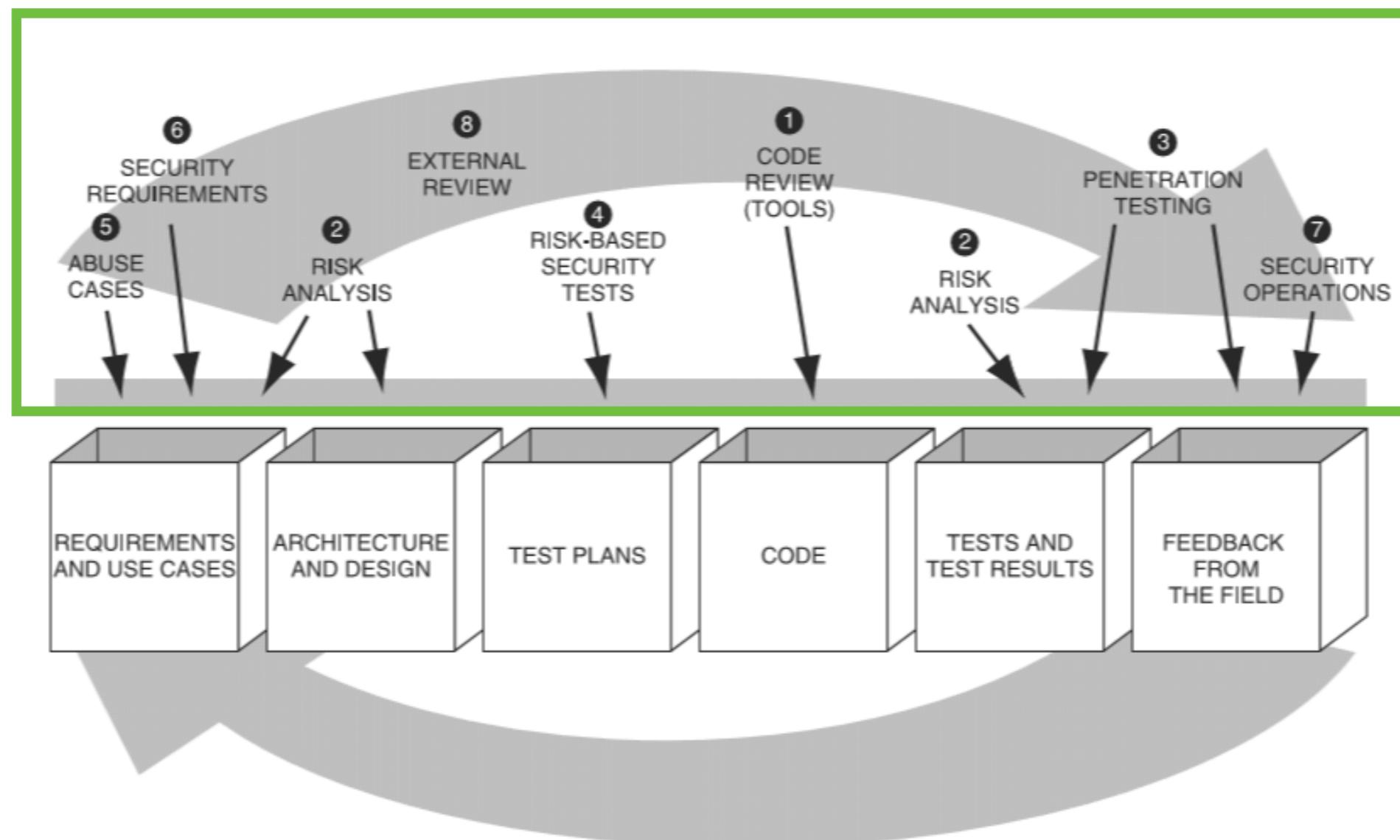
<https://checkmarx.com/glossary/a-secure-sdlc-with-static-source-code-analysis-tools/>



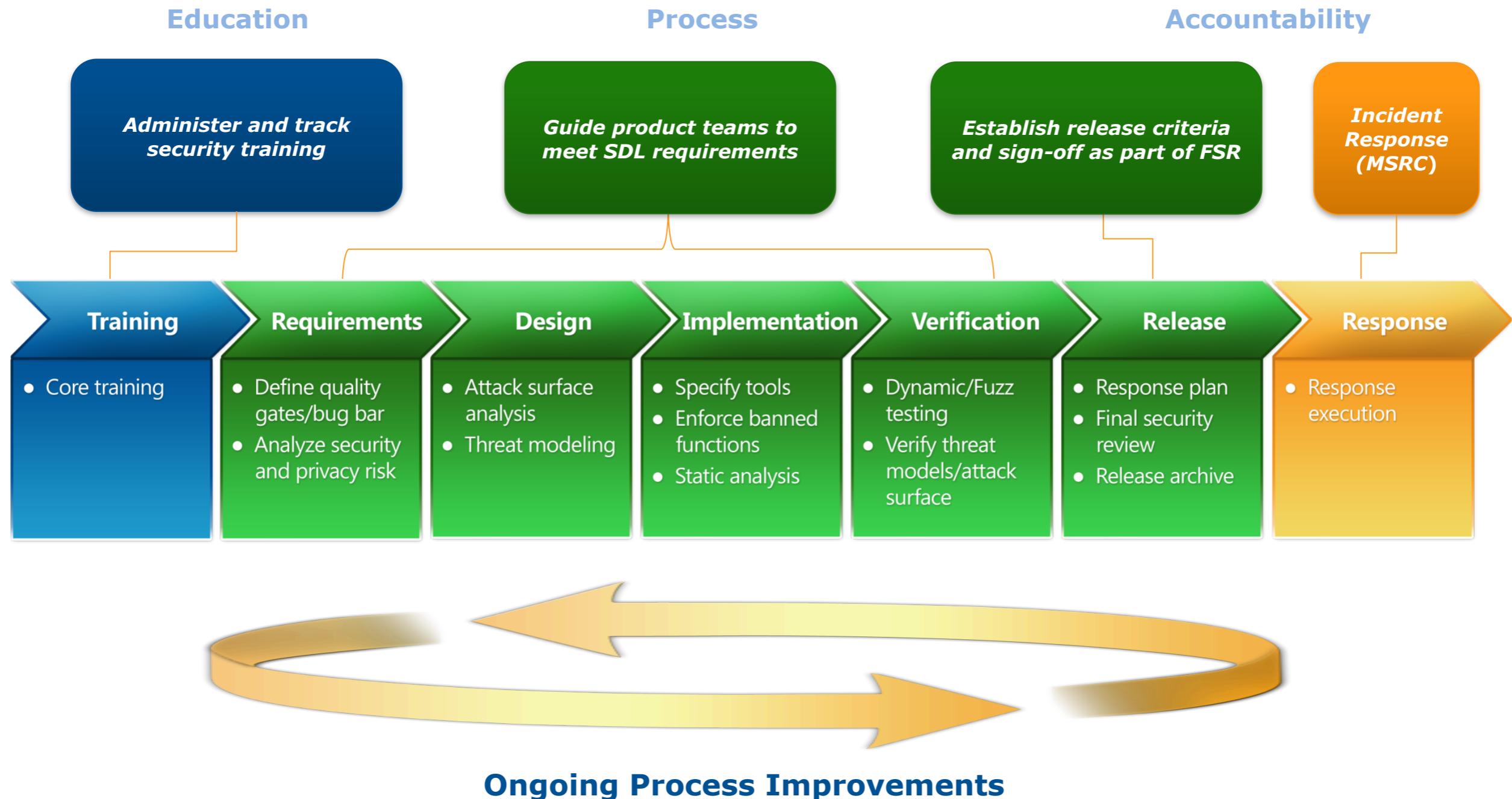
# Secure Software Development process

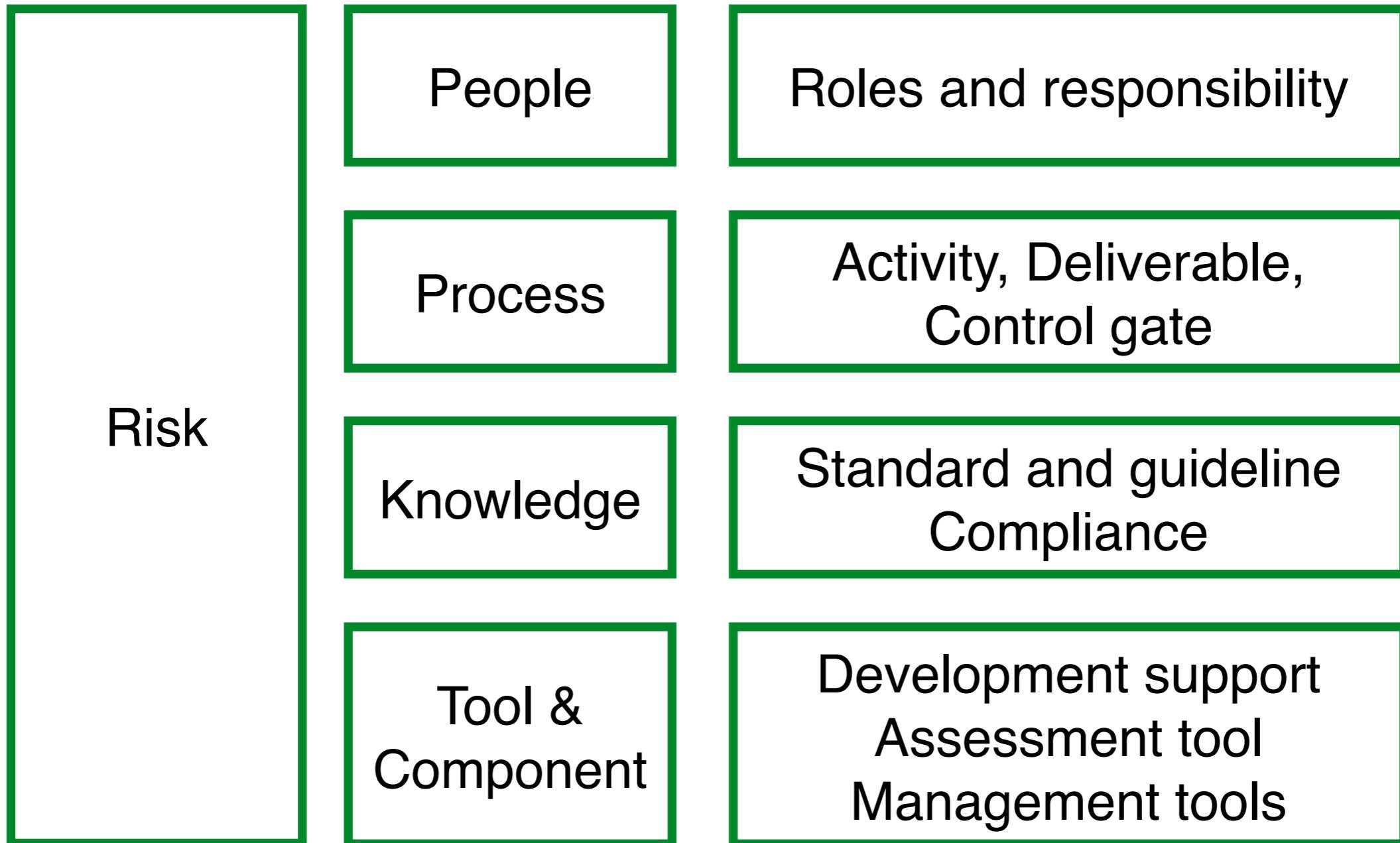


# Secure Software Development process



# Microsoft Security Development





# Pipeline to Deliver



## 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

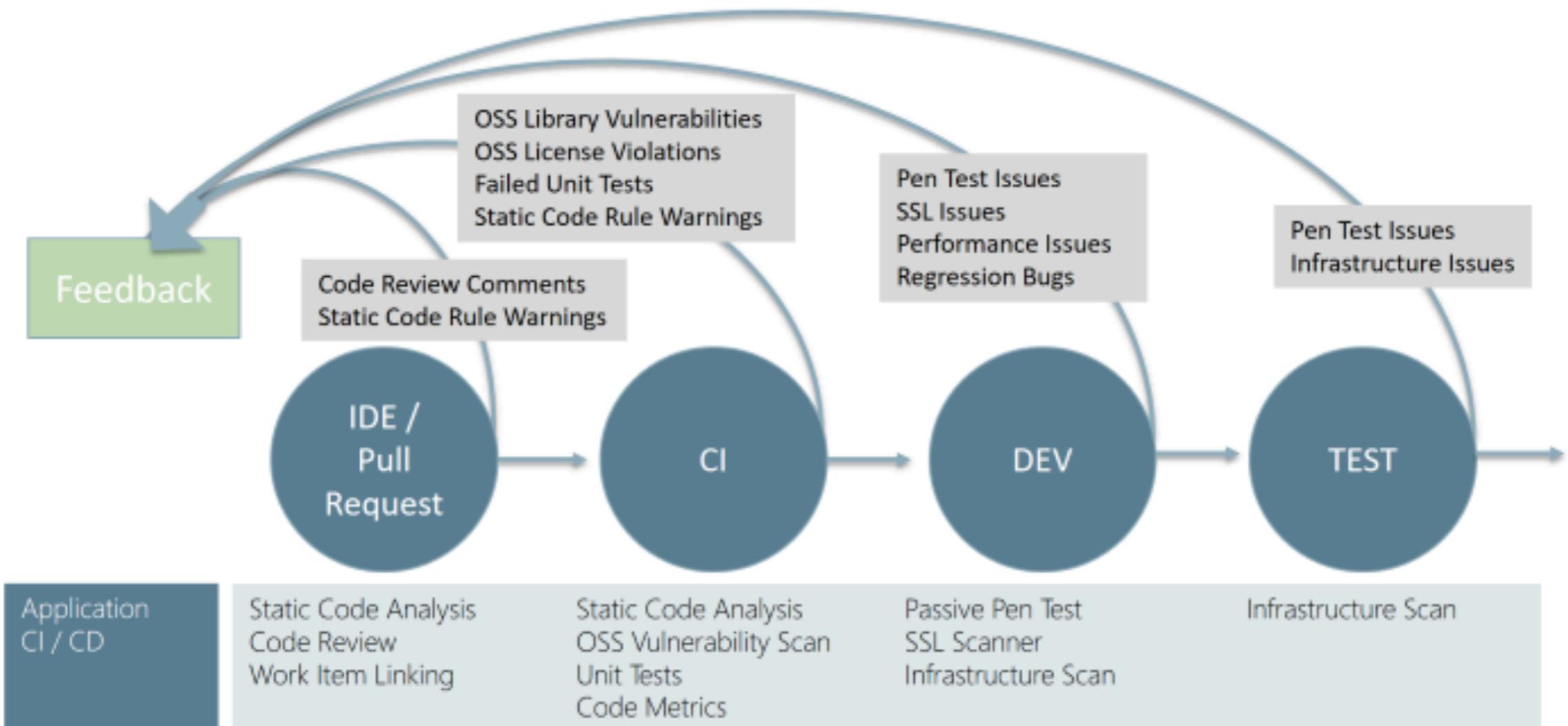
## Operate

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

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



# Feedback Loops

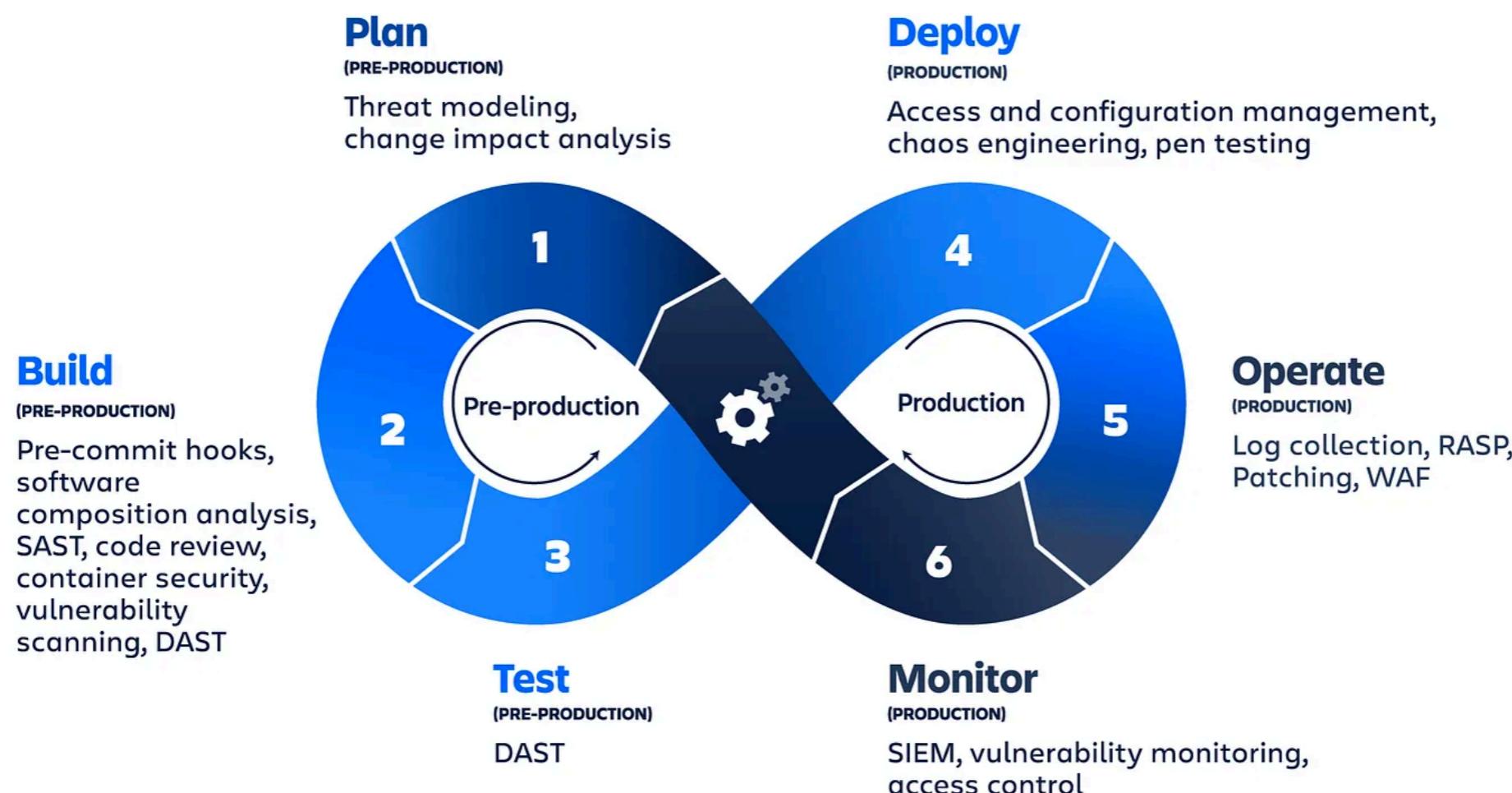


<https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/secure/devsecops-controls>



# DevSecOps

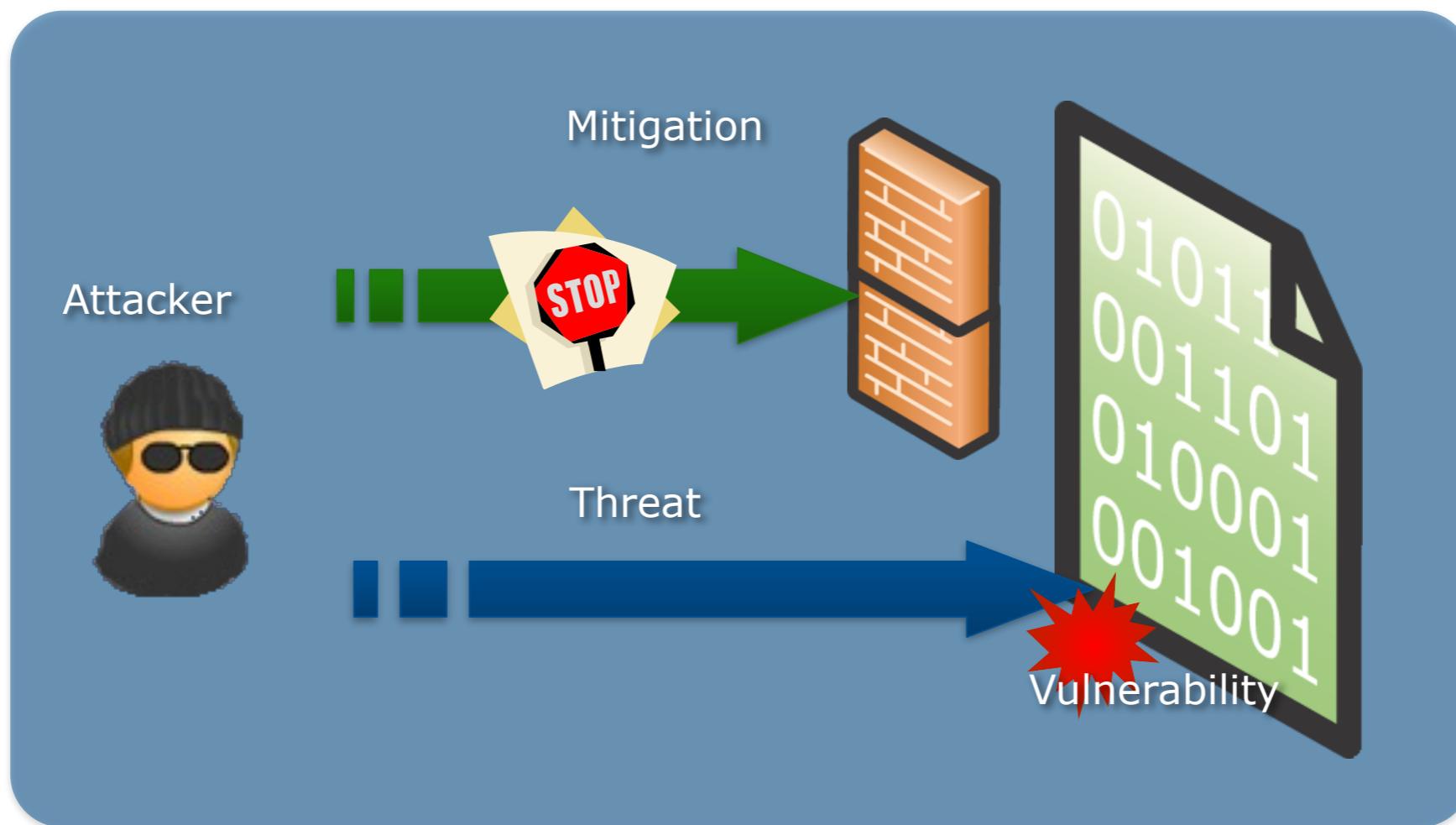
## DevSecOps



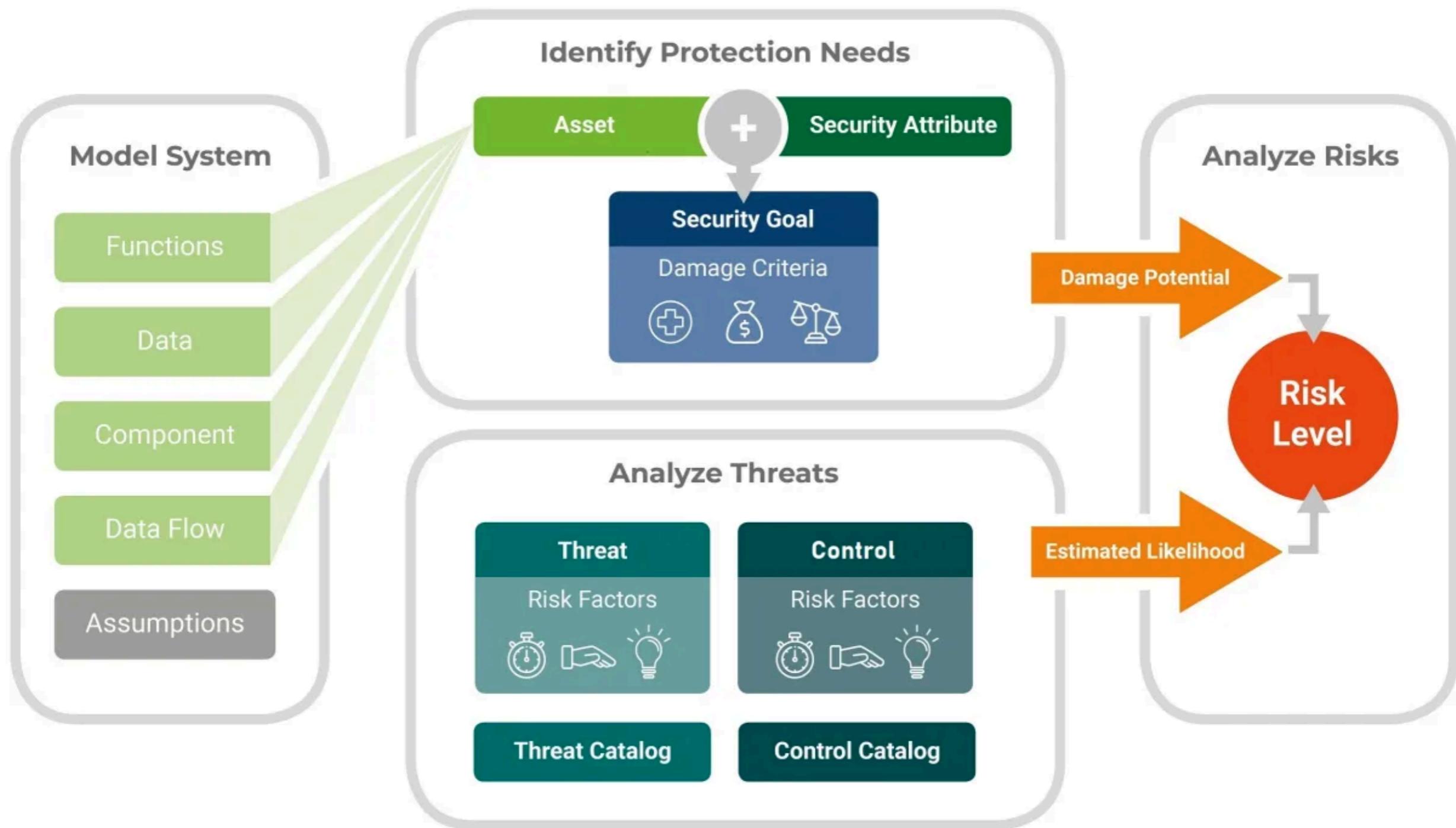
# Secure Design

## Threat Analysis

Secure software start with understanding the threats



# Threat Analysis



# Threat Modeling from OWASP

What are we working on ?

What can go wrong ?

What are we going to do about it ?

Did we do a good enough job ?

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



# Test Plan with Security ?

\



# Test Plan with Security ?

Security related test cases and scenarios

Test data related to security testing

Test tools required for security testing

Analysis of various test outputs from security tools



# Example of test scenarios

Password should be  
in encrypt format

System should not  
allow invalid users

Check data in web  
browser eg. Cookie,  
local storage



# Workshop with Login Page

## Sign In

Email address

Password

Remember me

**Submit**

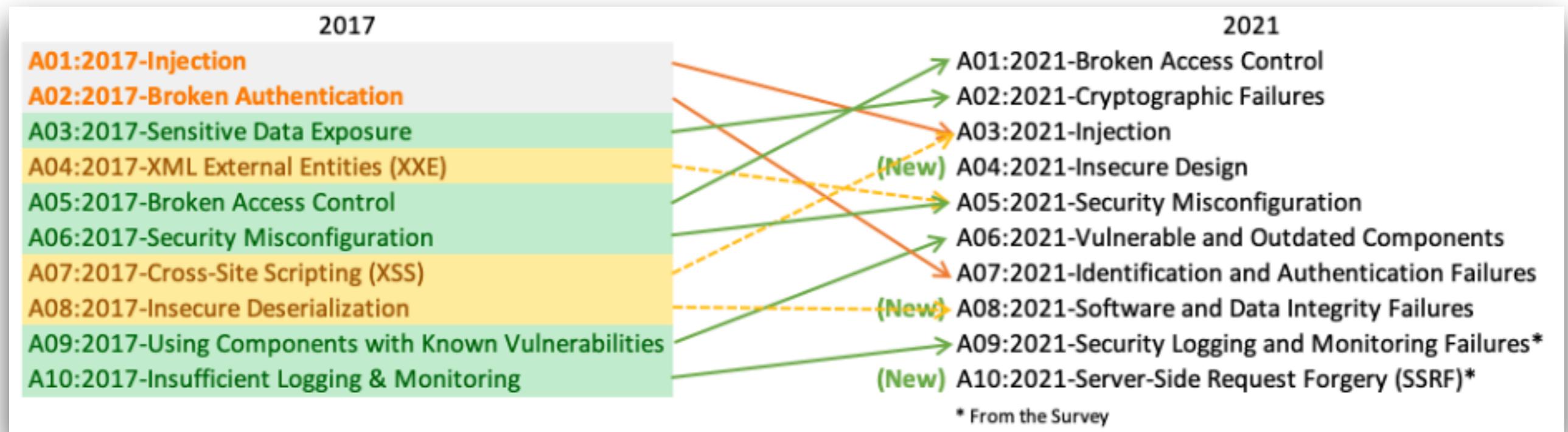
[Forgot password?](#)



# **Design test scenarios with security ?**



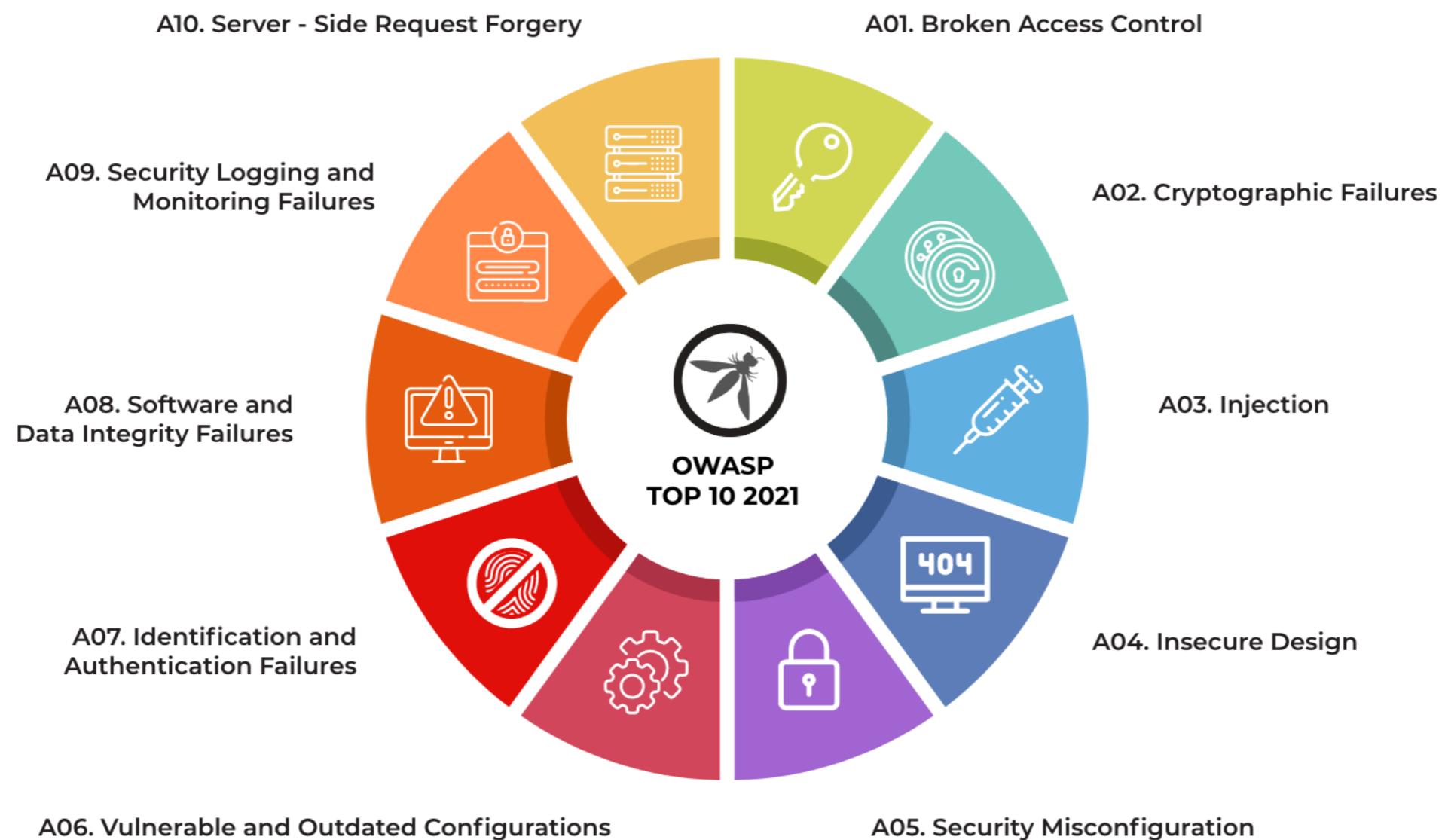
# Top 10 Web Application Security Risks



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



# Top 10 Web Application Security Risks



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



# What to test ?

HTTPS and data  
encryption

Data/SQL Injection

Cross-Site Scripting  
(XSS)

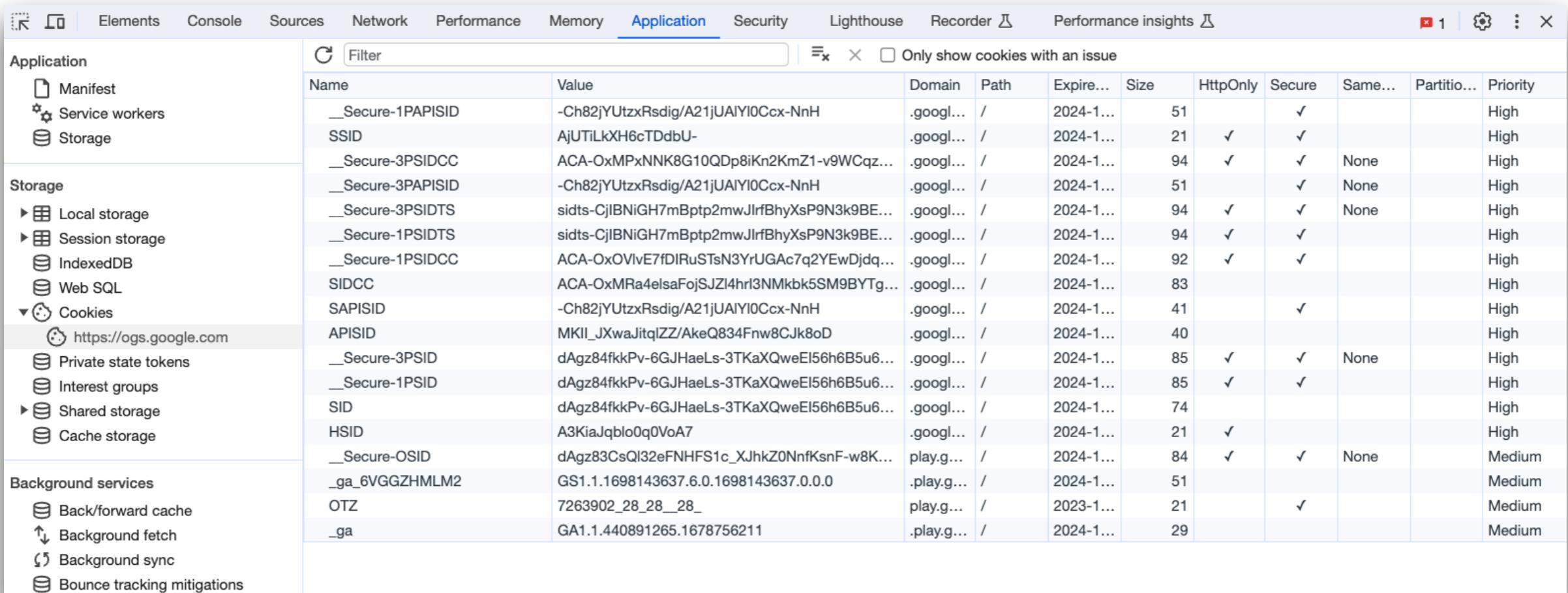
Brute-force attack

Secure cookie in  
browser, HttpOnly



# Google Chrome

## Inspect -> Application



The screenshot shows the Google Chrome DevTools interface with the "Application" tab selected. On the left, there's a sidebar with sections for Manifest, Service workers, Storage, Cookies, Private state tokens, Interest groups, Shared storage, Cache storage, and Background services. The "Cookies" section is expanded, showing a list of cookies for the domain https://ogs.google.com. The table has columns for Name, Value, Domain, Path, Expires, Size, HttpOnly, Secure, SameSite, Partition, and Priority. Most cookies have a priority of "High".

Name	Value	Domain	Path	Expires	Size	HttpOnly	Secure	SameSite	Partition	Priority
__Secure-1PAPISID	-Ch82jYUtzxRsdig/A21jUAIYI0Ccx-NnH	.googl...	/	2024-1...	51		✓			High
SSID	AjUTiLkXH6cTDdbU-	.googl...	/	2024-1...	21	✓	✓			High
__Secure-3PSIDCC	ACA-OxMPxNNK8G10QDp8iKn2KmZ1-v9WCqz...	.googl...	/	2024-1...	94	✓	✓	None		High
__Secure-3PAPISID	-Ch82jYUtzxRsdig/A21jUAIYI0Ccx-NnH	.googl...	/	2024-1...	51		✓	None		High
__Secure-3PSIDTS	sidts-CjIBNiGH7mBptp2mwJlrfBhyXsP9N3k9BE...	.googl...	/	2024-1...	94	✓	✓	None		High
__Secure-1PSIDTS	sidts-CjIBNiGH7mBptp2mwJlrfBhyXsP9N3k9BE...	.googl...	/	2024-1...	94	✓	✓			High
__Secure-1PSIDCC	ACA-OxOlvE7fDIRuSTsN3YrUGAc7q2YEwDjq...	.googl...	/	2024-1...	92	✓	✓			High
SIDCC	ACA-OxMRa4elsaFojSJZl4hrI3NMkbk5SM9BYTg...	.googl...	/	2024-1...	83					High
SAPISID	-Ch82jYUtzxRsdig/A21jUAIYI0Ccx-NnH	.googl...	/	2024-1...	41		✓			High
APISID	MKII_JXwajitqlZZ/AkeQ834Fnw8CJk8oD	.googl...	/	2024-1...	40					High
__Secure-3PSID	dAgz84fkkPv-6GJHaeLs-3TKaXQweEl56h6B5u6...	.googl...	/	2024-1...	85	✓	✓	None		High
__Secure-1PSID	dAgz84fkkPv-6GJHaeLs-3TKaXQweEl56h6B5u6...	.googl...	/	2024-1...	85	✓	✓			High
SID	dAgz84fkkPv-6GJHaeLs-3TKaXQweEl56h6B5u6...	.googl...	/	2024-1...	74					High
HSID	A3KiaJqblo0q0VoA7	.googl...	/	2024-1...	21	✓				High
__Secure-OSID	dAgz83CsQl32eFNHFS1c_XjhkZ0NnfKsnF-w8K...	play.g...	/	2024-1...	84	✓	✓	None		Medium
_ga_6VGGZHMLM2	GS1.1.1698143637.6.0.1698143637.0.0.0	.play.g...	/	2024-1...	51					Medium
OTZ	7263902_28_28_28_	.play.g...	/	2023-1...	21		✓			Medium
_ga	GA1.1.440891265.1678756211	.play.g...	/	2024-1...	29					Medium



# OWASP Application Security Verification Standard

## Input Validation, Sanitization and Encoding

### Control Objective

The most common web application security weakness is the failure to properly validate input coming from the client or the environment before directly using it without any output encoding. This weakness leads to almost all of the significant vulnerabilities in web applications, such as Cross-Site Scripting (XSS), SQL injection, interpreter injection, locale/Unicode attacks, file system attacks, and buffer overflows.

Ensure that a verified application satisfies the following high-level requirements:

- Input validation and output encoding architecture have an agreed pipeline to prevent injection attacks.
- Input data is strongly typed, validated, range or length checked, or at worst, sanitized or filtered.
- Output data is encoded or escaped as per the context of the data as close to the interpreter as possible.

With modern web application architecture, output encoding is more important than ever. It is difficult to provide robust input validation in certain scenarios, so the use of safer API such as parameterized queries, auto-escaping templating frameworks, or carefully chosen output encoding is critical to the security of the application.

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

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



# Workshop



# OWASP Juice Shop

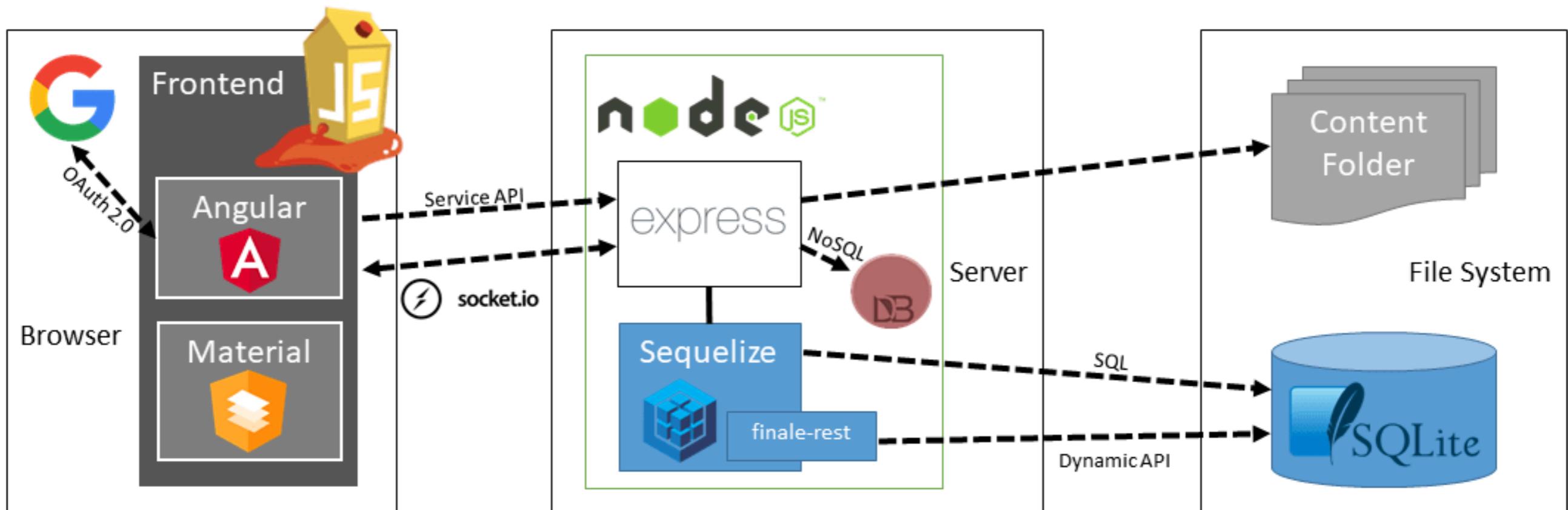
The screenshot shows a dark-themed web application interface for 'OWASP Juice Shop'. At the top, there's a header bar with a logo, the title 'OWASP Juice Shop', a search icon, an 'Account' link, and a language switcher ('EN'). Below the header, a button labeled 'All Products' is visible. The main content area displays a grid of six product cards and one artwork card:

- Apple Juice (1000ml)**: Price 1.99¤. Image shows a juice cup and an apple.
- Apple Pomace**: Price 0.89¤. Image shows a container of pomace and two apples.
- Banana Juice (1000ml)**: Price 1.99¤. Image shows a juice cup and a banana.
- Best Juice Shop Salesman Artwork**: Price 5000¤. Image shows a cartoon salesman holding a juice bottle with a green overlay that says 'Only 1 left'.
- Carrot Juice (1000ml)**: Price 2.99¤. Image shows a juice cup and a carrot.
- Eggfruit Juice (500ml)**: Price 8.99¤. Image shows a juice cup and an eggfruit.

<https://demo.owasp-juice.shop/#/>



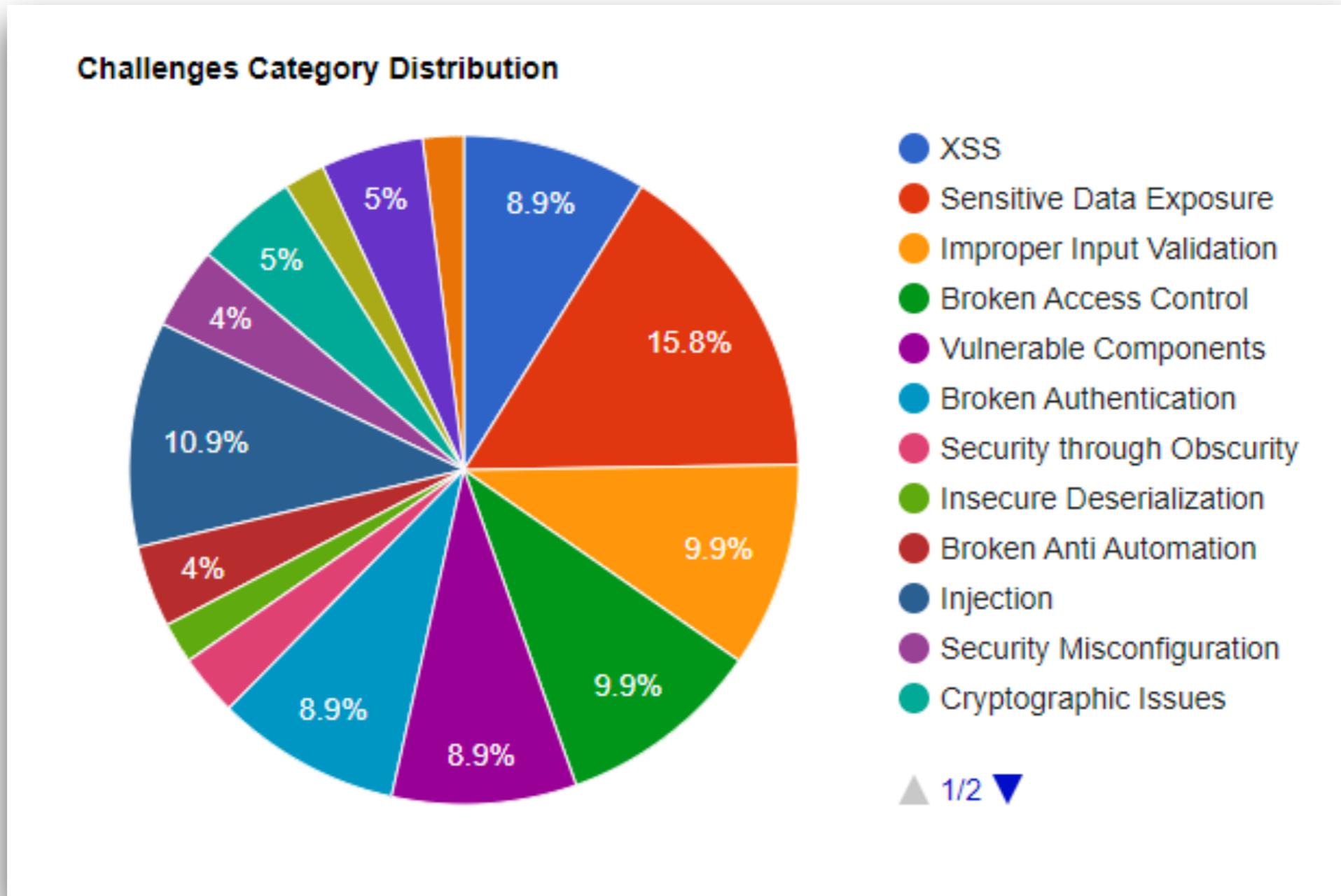
# OWASP Juice Shop



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



# OWASP Juice Shop covers all vulnerabilities



# Score Board

The screenshot shows the OWASP Juice Shop Score Board page. At the top, there's a navigation bar with a menu icon, the logo 'OWASP Juice Shop', a search icon, an account icon, and language settings ('EN'). Below the header, there are two progress bars: 'Score Board 41%' and 'Coding Score 0%'. Underneath these are six star icons representing different challenges, each with a percentage completion: 13/14 (1), 12/14 (2), 12/23 (3), 5/25 (4), 1/18 (5), and 0/12 (6). To the right of these are buttons for 'Show all', 'Show solved', 'Show tutorials only', 'Show unavailable', and two download/copy icons. Below the challenges is a horizontal menu with categories: Broken Access Control, Broken Anti Automation, Broken Authentication, Cryptographic Issues, Improper Input Validation, Injection, Insecure Deserialization, Miscellaneous, Security Misconfiguration, Security through Obscurity, Sensitive Data Exposure, Unvalidated Redirects, Vulnerable Components, XSS, and XXE. A 'Hide all' button is also present. At the bottom of the page, there's a note about the old score board being slow or confusing, followed by a link to the new one. The main content area displays a table of challenges:

Name	Difficulty	Description	Category	Tags	Status
Bonus Payload	★	Use the bonus payload <iframe width="100%" height="166" src="url=https%3A//api.soundcloud.com/tracks/771984076&colorXSS</iframe> in the DOM XSS challenge.		Shenanigans Tutorial	<input checked="" type="checkbox"/> solved
Bully Chatbot	★	Receive a coupon code from the support chatbot.	Miscellaneous	Brute Force Shenanigans	<input checked="" type="checkbox"/> solved
Confidential Document	★	Access a confidential document.	Sensitive Data Exposure	Good for Demos	<input checked="" type="checkbox"/> solved
DOM XSS	★	Perform a DOM XSS attack with <iframe src="javascript:alert(`xss`)">.	XSS	Good for Demos Tutorial	<input checked="" type="checkbox"/> solved

<https://demo.owasp-juice.shop/#/score-board>



# Try to login

Login

Email \*

Password \*

Forgot your password?

Log in

Remember me

or

G Log in with Google

Not yet a customer?

<https://demo.owasp-juice.shop/#/login>



# Test cases

Group of test	Test cases name	Test data
Success case	Valid email and password	
Failure case	Wrong email	
	Wrong password	
	Wrong both email and password	
Edge case	Empty email and password	
	Maximum length of email and password	
Security case	SQL injection in email and password	' OR '1'='1 --
	XSS in email and password	<script>alert('XSS')</script>@example.com



# Try to login with SQL Injection

The screenshot shows a dark-themed login interface. At the top is a "Login" title. Below it are two input fields: "Email \*" and "Password \*". The "Password" field includes a visibility icon. A "Forgot your password?" link is located below the password field. In the center is a "Log in" button with a right-pointing arrow icon. To its left is a "Remember me" checkbox. Below the "Log in" button is a horizontal line with the word "or" in the center. To the right of the "or" line is a green "G" icon followed by the text "Log in with Google". At the bottom of the form is a link "Not yet a customer?".

<https://demo.owasp-juice.shop/#/login>



# Example of Inputs

Input	Comments
OR '1'='1	
admin' --	-- = comment in sql
; DROP TABLE users; --	
admin' OR '1'='1	
admin' or '1'='1'--	
><script>alert('Injected')</script>	Code Injection
; DROP TABLE users; --	



# Try to login with SQL Injection

Login

Email \* **admin' or '1'='1'--**

Password \*

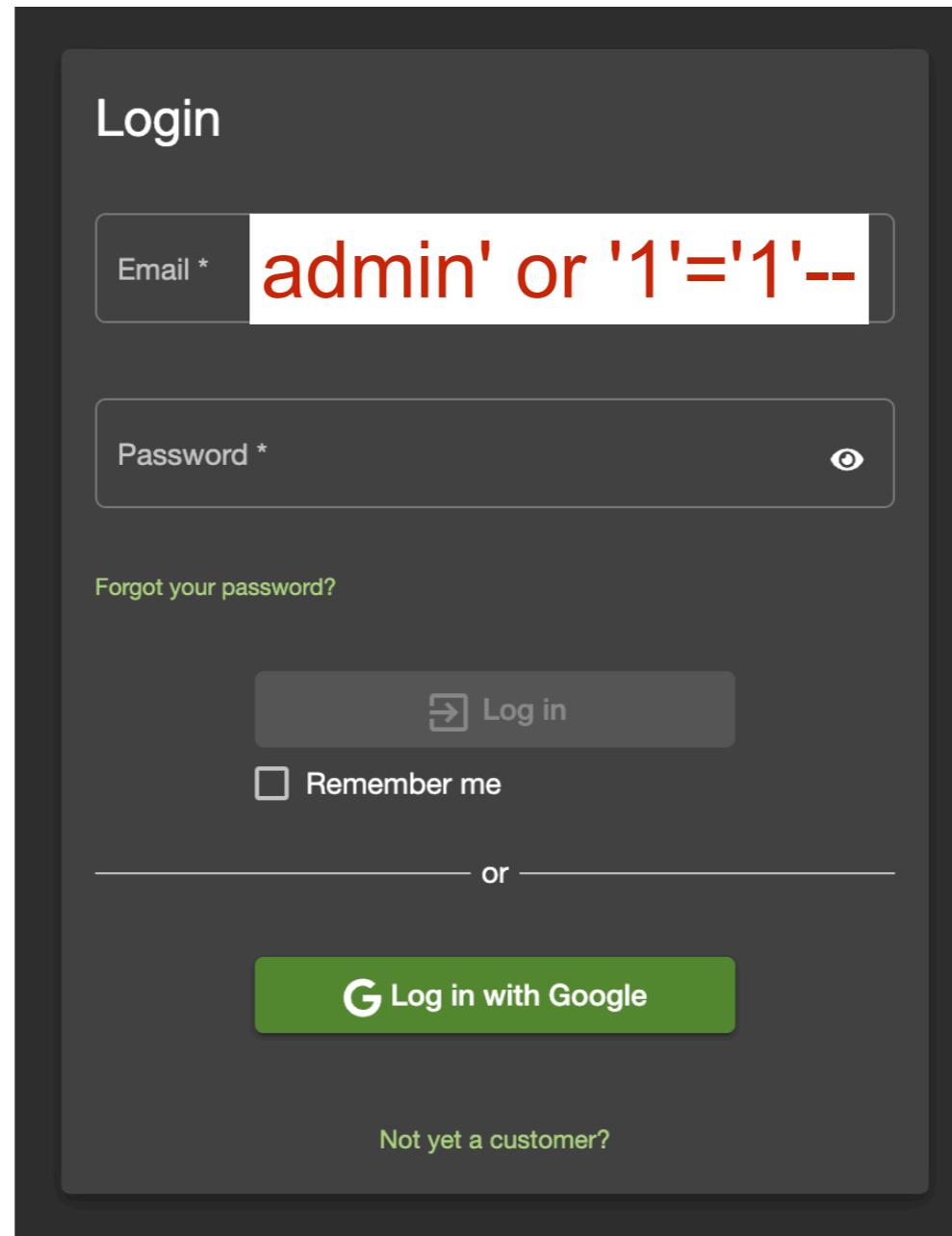
[Forgot your password?](#)

Remember me

or

**G** Log in with Google

Not yet a customer?



<https://demo.owasp-juice.shop/#/login>



# Register

User Registration

Email \*

Password \*

! Password must be 5-40 characters long. 0/20

Repeat Password \*

! This cannot be changed later! 0/40

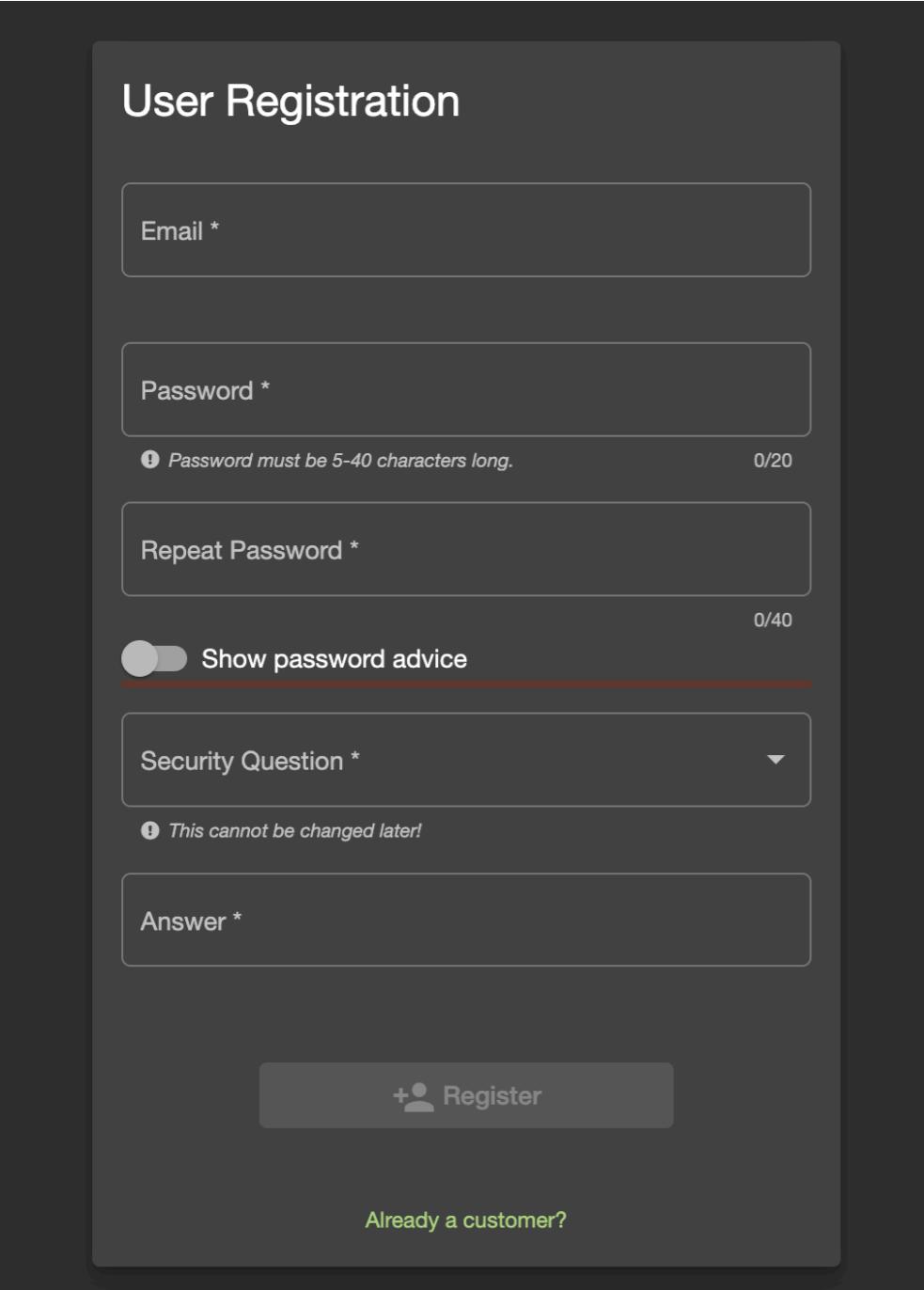
Show password advice

Security Question \*

Answer \*

+ Register

Already a customer?



/api/Users/

?

API

<https://demo.owasp-juice.shop/#/register>



# Secure User Login ?

Password hashing

Salt the hash

Use HTTPS

Account lock

2FA

Password policy

Secure cookies

Avoid information leak



# Tools

 PortSwigger

LOGIN

Products ▾ | Solutions ▾ | Research | Academy | Support ▾ | ⚙

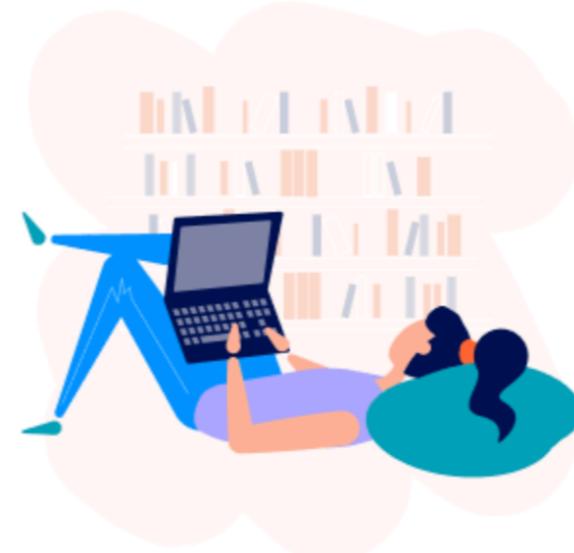
# Burp Suite Community Edition

Start your web security testing journey for free - download our essential manual toolkit.

Enter your email to download

⬇ DOWNLOAD

Go straight to downloads →

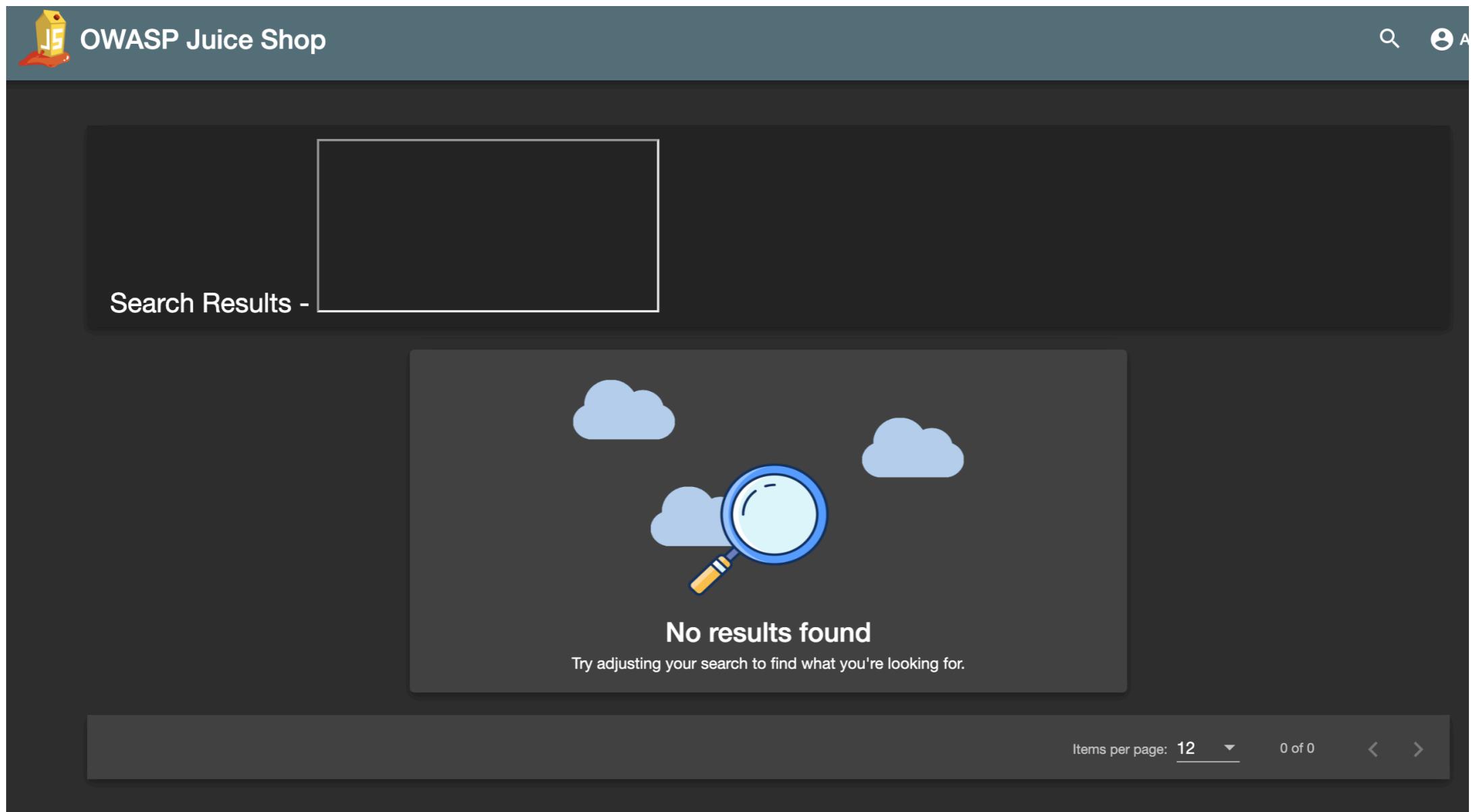


<https://portswigger.net/burp/communitydownload>



# Cross Site Scripting (XSS)

```
<iframe src="javascript:alert(`data all over this screen that wasnt planned`)">
```



<https://demo.owasp-juice.shop/#/search?q=apple>



# Feedback

## Customer Feedback

Author  
anonymous

Comment \*

Max. 160 characters 0/160

Rating

CAPTCHA: What is  $9+3*7$  ?

Result \*

> Submit

<https://demo.owasp-juice.shop/#/contact>



# Broken Access Control

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



# Broken Access Control

Users can access resources that they are not supposed to be able to

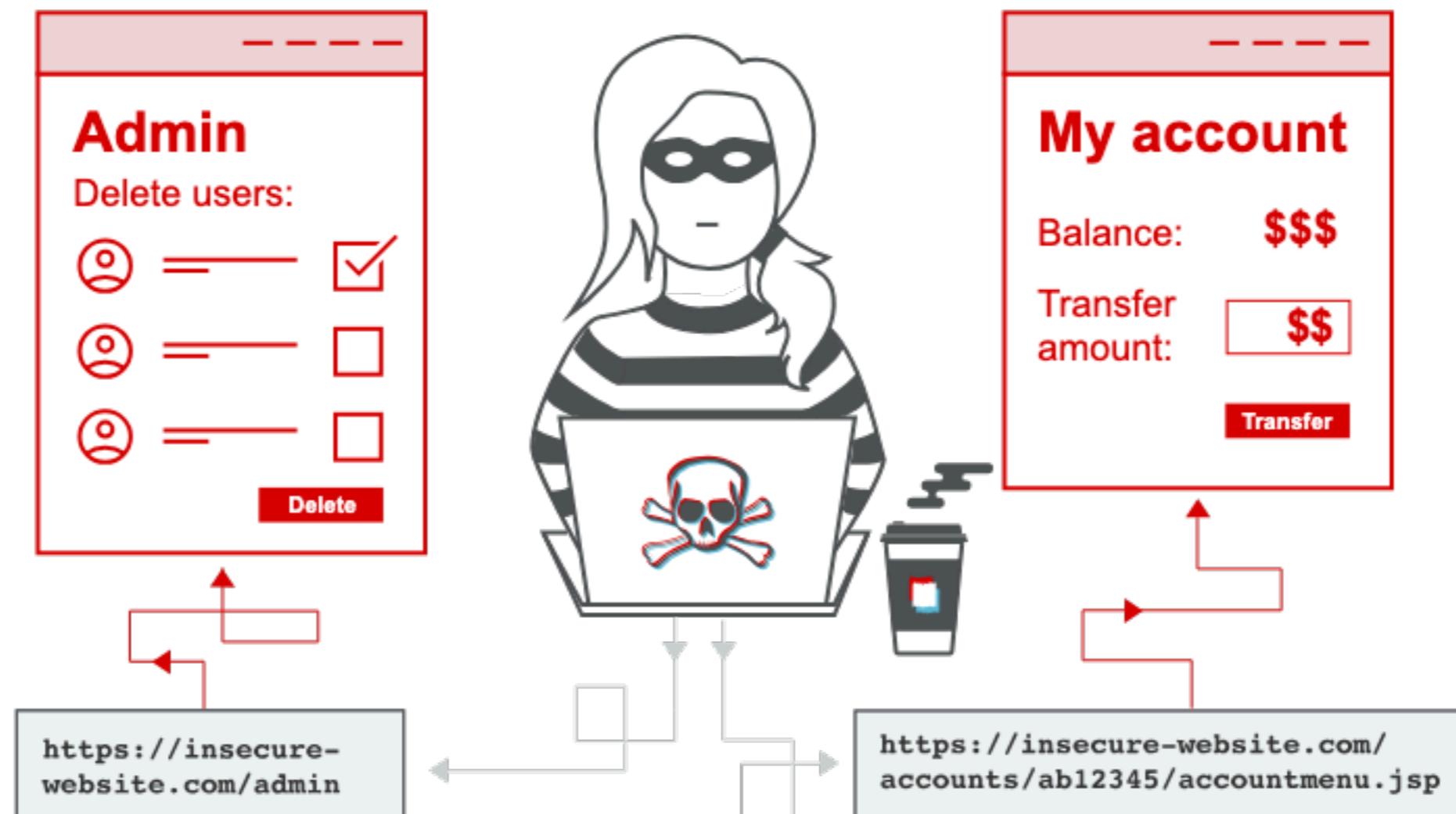
Authentication

Session  
management

Access control



# Broken Access Control



# Broken Access Control

In juice shop

Forged review

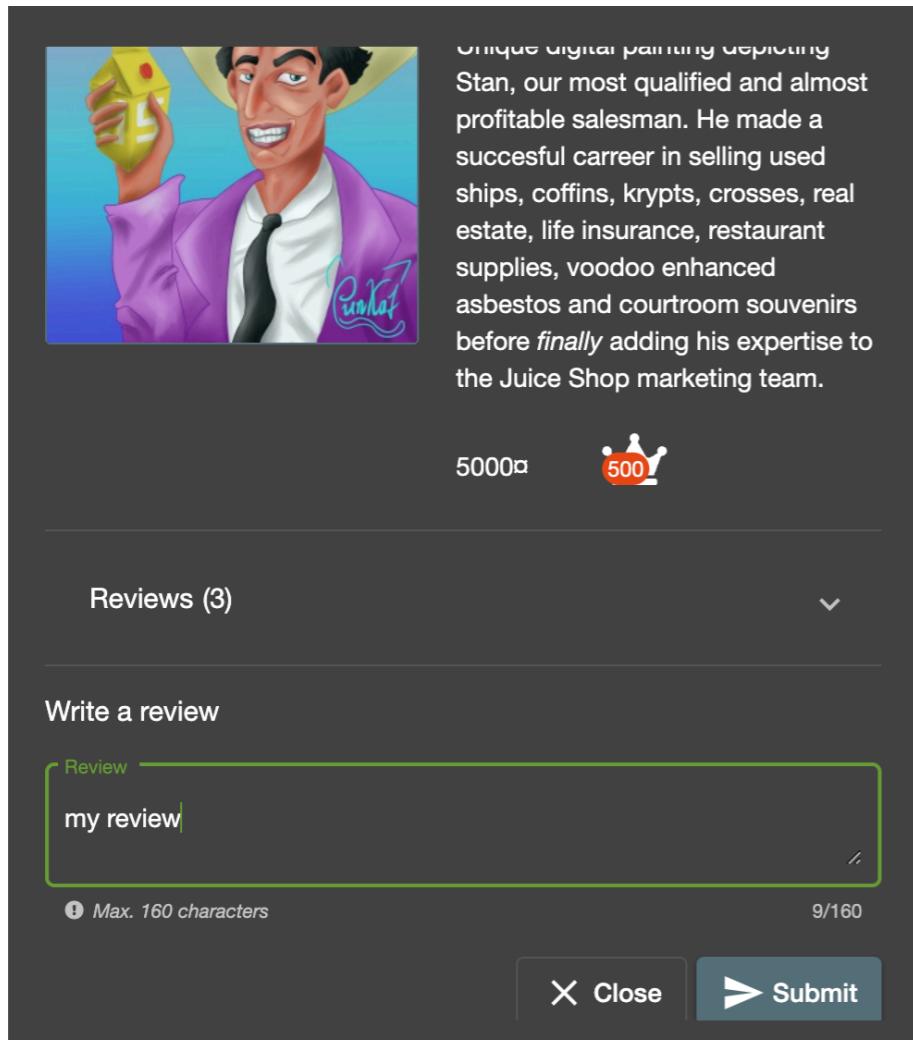
Forged  
feedback

Add product to  
other basket



# Forged review

User can write a review.



rest/products/24/reviews



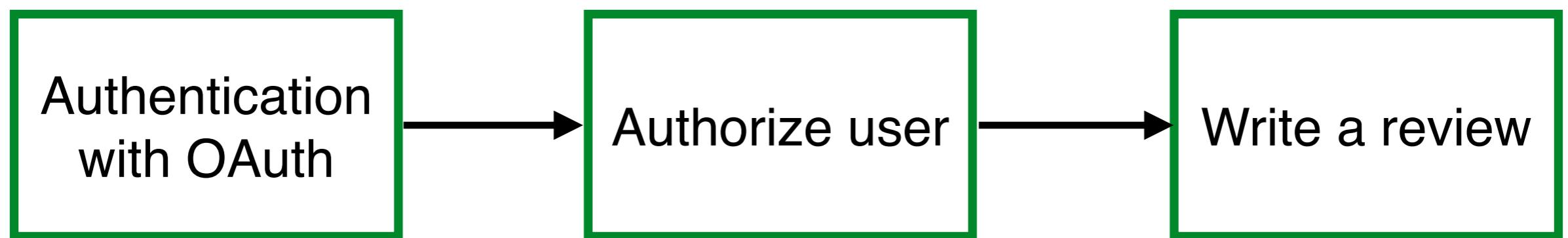
?

API

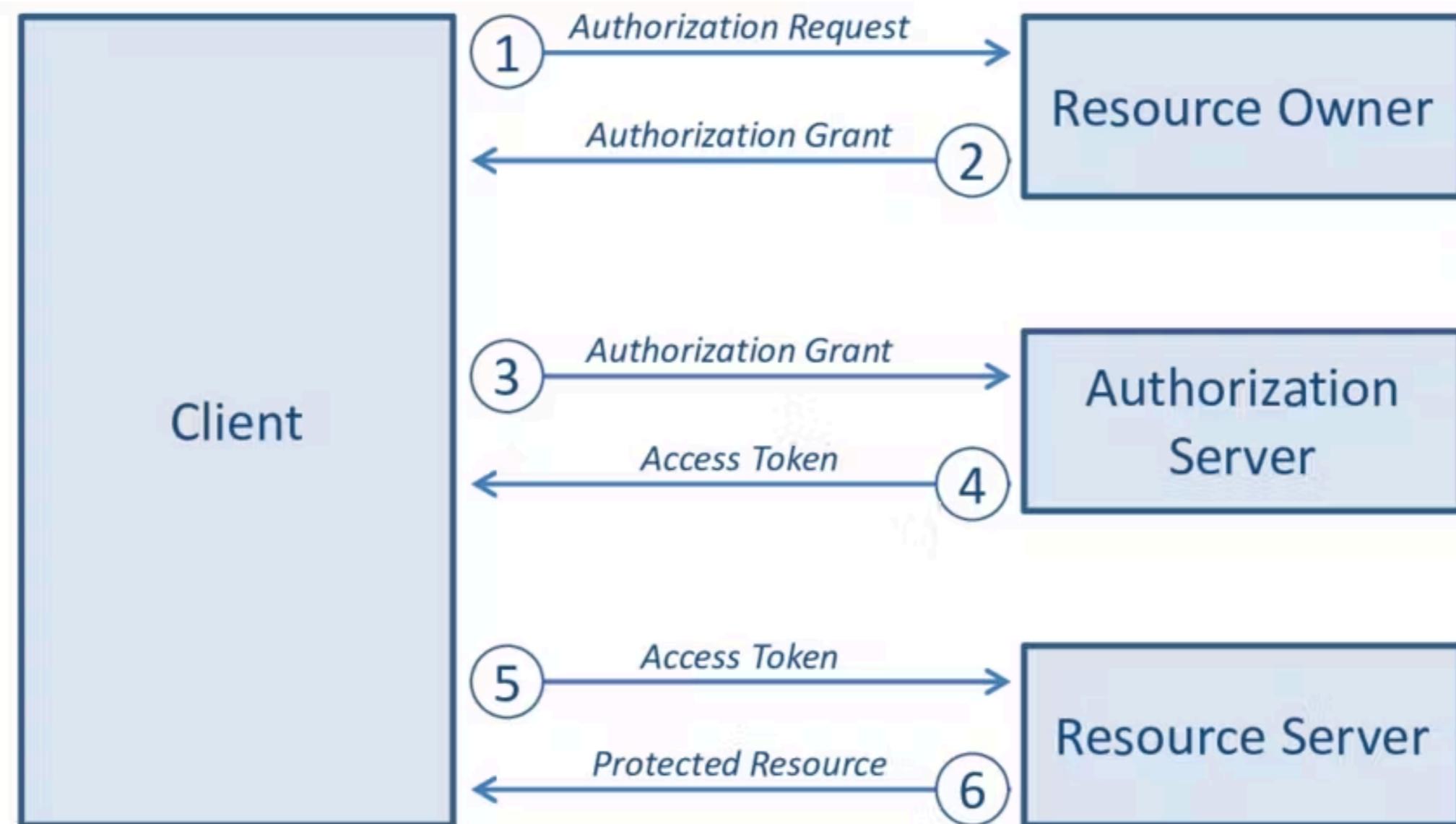


# How to prevent ?

User can write a review.



# OAuth 2.0



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

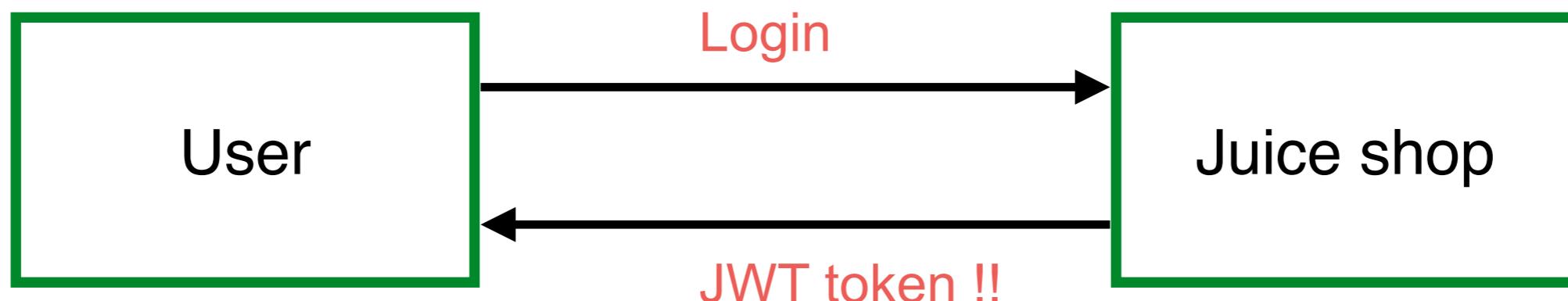


# Unsigned JWT Token



# Forged Signed JWT

JWT Token !!



[https://pwning.owasp-juice.shop/companion-guide/latest/part2/vulnerable-components.html#\\_forge\\_an\\_almost\\_properly\\_rsa\\_signed\\_jwt\\_token](https://pwning.owasp-juice.shop/companion-guide/latest/part2/vulnerable-components.html#_forge_an_almost_properly_rsa_signed_jwt_token)



# Log in to system

email=jwtn3d@juice-sh.op  
password=123456



[https://pwning.owasp-juice.shop/companion-guide/latest/part2/vulnerable-components.html#\\_forge\\_an\\_almost\\_properly\\_rsa\\_signed\\_jwt\\_token](https://pwning.owasp-juice.shop/companion-guide/latest/part2/vulnerable-components.html#_forge_an_almost_properly_rsa_signed_jwt_token)



# Open JWT Token

**Encoded** PASTE A TOKEN HERE

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MzYsInVzZXJuYW1lIjoiIiwiZW1haWwiOiJy c2FfbG9yZEBqdWljZS1zaC5vcCIsInBhc3N3b3JkIjoiODI3Y2NiMGV1YThhNzA2YzRjMzRhMTY40T FmODR1N2IiLCJyb2xIiJoiY3VzdG9tZXIiLCJkZ Wx1eGVUb2tlbiI6IiIsImxhc3RMb2dpbkIwIjoi MC4wLjAuMCIsInByb2ZpbGVJbWFnZSI6Ii9hc3N1dHMvcHVibGljl2ltYWdlcy91cGxvYWRzL2R1Zm F1bHQuc3ZnIiwidG90cFN1Y3JldCI6IiIsImlzQ WN0aXZ1Ijp0cnV1LCJjcmVhdGVkQXQiOiIyMDIz LTExLTE2IDEy0jA40jA1Ljk3MCArMDA6MDAiLCJ1cGRhdGVkQXQiOiIyMDIzLTExLTE2IDEy0jA40jA1Ljk3MCArMDA6MDAiLCJkZWxldGVkQXQiOm51bGx9LCJpYXQiOjE3MDAxMzY00Th9.G0EIIm8kSikB1ji9VPbgWgME4sZuBIJi2HxUc1-4RXEGMXyqbgmtVireNahJvFr- m3HogkZyIYgKvccayrHc0j6e0TR3-8lGhny70vxG-qMQDFcKmJ0AZW-qro3K290RyC5k-xW0ibk8rK3LWYkdsG9cvaL8UVUkknLfUEe9t8E
```

**Decoded** EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE
{ "typ": "JWT", "alg": "RS256" }
PAYLOAD: DATA
{ "status": "success", "data": { "id": 36, "username": "", "email": "rsa_lord@juice-sh.op", "password": "827ccb0eea8a706c4c34a16891f84e7b", "role": "customer", "deluxeToken": "", "lastLoginIp": "0.0.0.0", "profileImage": "/assets/public/images/uploads/default.svg", "totpSecret": "", "isActive": true, "createdAt": "2023-11-16 12:08:05.970 +00:00", "updatedAt": "2023-11-16 12:08:05.970 +00:00", "deletedAt": null }, "iat": 1700136498 }
VERIFY SIGNATURE

<https://jwt.io/>



# Create Unsigned JWT Token !!

Update alg="none"

**Header.Payload.**

<https://gchq.github.io/CyberChef/>



# JSON Web Token Attack

The screenshot shows the Burp Suite Community Edition interface. The top navigation bar includes Dashboard, Target, Proxy, Intruder, Repeater, Collaborator, Sequencer, Decoder, Comparer, Logger, Organizer, Extensions (which is selected), and Learn. Below the navigation is a status bar indicating 'Total estimated system impact: None'. The main area is titled 'BApp Store' and contains a table of installed extensions. The table columns are Name, Installed, Rating, Popularity, Last updated, System imp..., and Detail. The 'JSON Web Token Attacker' extension is highlighted with a blue selection bar at the bottom of the list. To the right of the table, there is a detailed view of the 'JSON Web Token Attacker' extension, including its name ('JOSEPH - JavaScript Object Signing and Encryption Pentesting Helper'), description ('This extension helps to test applications that use JavaScript Object Signing and Encryption, including JSON Web Tokens.'), features (Recognition and marking, JWS/JWE editors, (Semi-)Automated attacks, Bleichenbacher MMA, Key Confusion (aka Algorithm Substitution), Signature Exclusion, Base64url en-/decoder, Easy extensibility of new attacks), estimated system impact (Overall: Medium), author (Dennis Detering), version (1.0.2a), source (<https://github.com/portswigger/json-web-token-attacker>), update date (04 Feb 2022), rating (4 stars), popularity (Medium), and an 'Install' button.



# Security Test Tools

Static

Dynamic

<https://github.com/GoVanguard/main-security-testing-tools>



# **Static Analysis Security Testing**



# Static Analysis Security Testing

Dev, Ops

Code and Config

Lint

Dependency  
Check

Static code  
Analysis

Security  
Code Scan

Detect secret

Detect vulnerable  
components



# Lint Tools

Integrate tools with Text editor or IDE

The screenshot shows the Visual Studio Marketplace interface. At the top, there's a navigation bar with links for Visual Studio, Visual Studio Code, Azure DevOps, Subscriptions, and options to Build your own and Publish extensions. A sign-in link and a search icon are also present. Below the navigation, a search bar contains the word "lint". The main area displays 514 results for "lint", showing various extension cards. The first row includes "C/C++ Advanced Lint", "Sass Lint (deprecated)", "GLSL Lint", "Proto Lint", "openapi-lint", and "scss-lint". The second row includes "GLSL Lint (deprecated)", "lua lint", "Coffee Lint", "Haml Lint", "cpp-check-lint", and "Slim Lint". Each card provides information such as the extension name, developer, download count, description, rating, and status (e.g., FREE).

Extension	Developer	Downloads	Description	Rating	Status
C/C++ Advanced Lint	Joseph Benden	471K	An advanced, modern, static analysis extension for C/C++ that supports a number of...	★★★★★	FREE
Sass Lint (deprecated)	glen-84	308K	Sass Lint for Visual Studio Code	★★★★★	FREE
GLSL Lint	DanielToplak	105K	Linting of GLSL shader code	★★★★★	FREE
Proto Lint	Plex Systems	111K	Lint your protobufs using protolint	★★★★★	FREE
openapi-lint	mermade	80.4K	OpenAPI (Swagger) 2.0/3.0.x intellisense, validator, linter, converter, resolver	★★★★★	FREE
scss-lint	Adam Walzer	71K	a scss-lint extension for vscode	★★★★★	FREE
GLSL Lint (deprecated)	CADENAS GmbH	83.5K	Linting of GLSL shader code	★★★★★	FREE
lua lint	satoren	23.9K	Static error checker for Lua	★★★★★	FREE
Coffee Lint	Kaiyuan Liu	20.1K	CoffeeScript linter for VS Code.	★★★★★	FREE
Haml Lint	aki77	19.3K	Haml Lint for Visual Studio Code	★★★★★	FREE
cpp-check-lint	QiuminGe	21.6K	cppcheck cpplint	★★★★★	FREE
Slim Lint	Ali Ariff	11.9K	Slim Linter for Visual Studio Code	★★★★★	FREE

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



# Source Code Analysis Tools

Help to analyze source code or compiled code to find security flaws

Scalable

Identified well-known vulnerability (Injection, Buffer overflow)

Output helps developer

Authentication problem

Access control issue

Insecure use of cryptography

High number of false positive

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



# Static Code Analysis

SonarQube is popular tool

The screenshot shows the SonarQube website homepage. At the top, there's a dark purple header with the Sonar logo and navigation links for Solutions, Products, Resources, and Company. Below the header, the main navigation includes 'sonarqube' (with a blue gear icon), Deployment, What's New, Roadmap, Documentation, Download, and a prominent 'START FREE TRIAL' button. The main content area features a large red banner with the text 'SELF-MANAGED. SONARQUBE.' and 'clean code for teams and enterprises with {SonarQube}'. Below this, a paragraph explains the solution's purpose: '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.' Two buttons are present: 'START FREE TRIAL -->' and 'WHAT IS SONARQUBE? >'. To the right of the banner, a sample analysis report card is displayed, showing a 'Passed' status with a green checkmark. It includes sections for Reliability (0 Bugs), Maintainability (0 Code Smells), Security (0 Vulnerabilities), Security Review (0 Security Hotspots), Coverage (76% Coverage, 21k lines), and Duplications (0.0% Duplications, 46k lines). A small note indicates 'New Code: since about 1 month ago'.

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



# Security Code Scan

## For .NET project

Visual Studio | Marketplace

Sign in Search

Visual Studio > Tools > Security Code Scan (for VS2019 and newer)

**Security Code Scan (for VS2019 and newer)**

JaroslavLobacevski | 17,091 installs | ★★★★★ (0) | Free

Security static code analyzer for .NET

[Download](#)

[Overview](#) [Rating & Review](#)

**Security static code analyzer for .NET**

[Website](#)

- Detects various [security vulnerability patterns](#): SQL Injection, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), XML eXternal Entity Injection (XXE), etc.
- Basic intraprocedural taint analysis for input data.
- Analyzes .NET and [.NET Core](#) projects in a background (IntelliSense) or during a build.
- Continuous Integration (CI) with [GitHub action](#) or through [MSBuild](#).
- Works with Visual Studio 2019 or higher. Visual Studio [Community](#), Professional and Enterprise editions are supported. Other editors that support Roslyn based analyzers like Rider or OmniSharp should work too.
- [Open Source](#)

**Categories**

.Tools .Coding .Security

**Tags**

.NET Analyzer Analyzers CSRF Injection  
OWASP Roslyn Security Security Code Scan  
Security.Code.Scan SecurityCodeScan SQLi  
Static Code Analysis Vulnerability XSS XXE

**Works with**

Visual Studio 2019, 2022 (amd64)

**Resources**

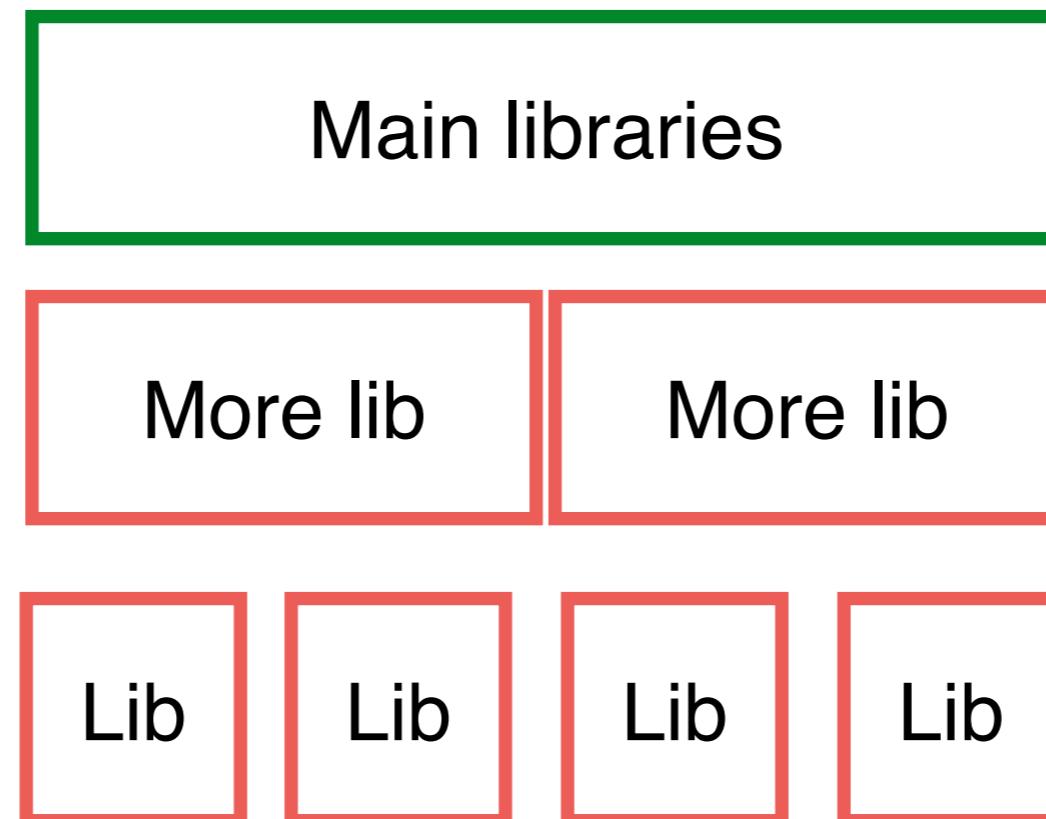
[License](#)

<https://marketplace.visualstudio.com/items?itemName=JaroslavLobacevski.SecurityCodeScanVS2019>



# Dependency Check

Check your libraries/dependencies in project

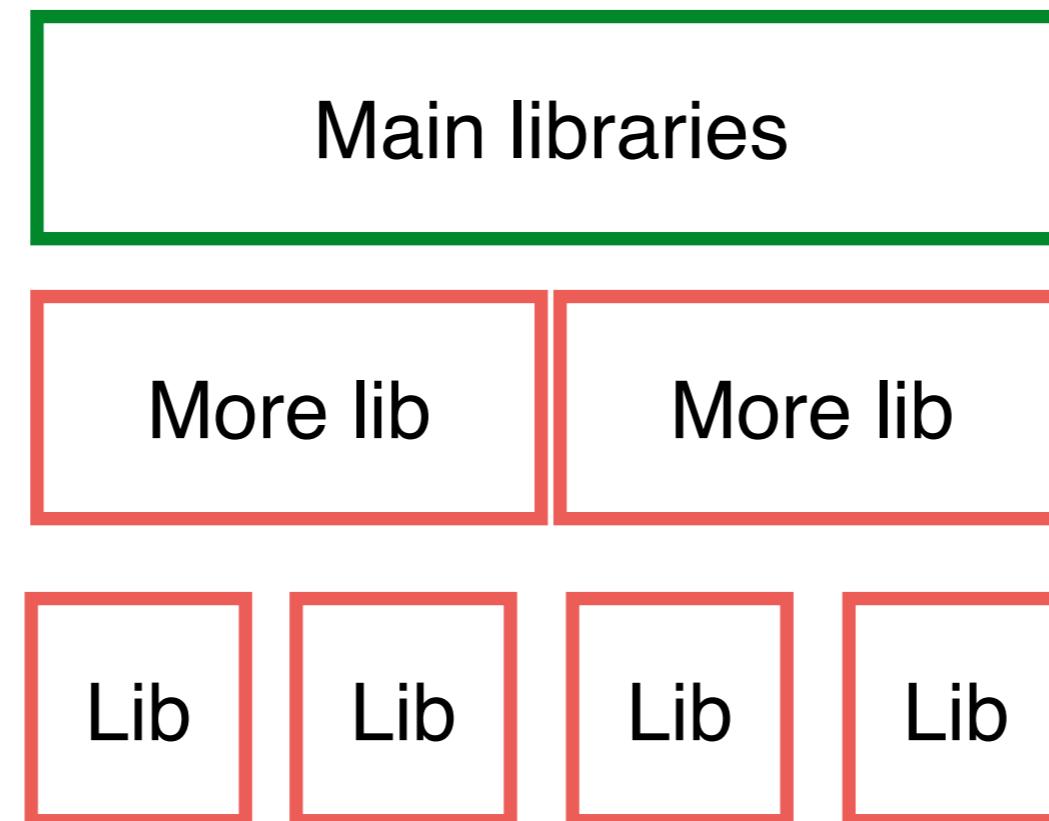


<https://owasp.org/www-project-dependency-check/>



# Dependabot from GitHub

Check your libraries/dependencies in GitHub project



<https://docs.github.com/en/code-security/getting-started/dependabot-quickstart-guide>



# Example with Log4Shell



<https://www.lunasec.io/docs/blog/log4j-zero-day/>



# Dynamic Analysis Security Testing



# OWASP ZAP

The screenshot shows the official website for Zed Attack Proxy (ZAP). At the top, there is a navigation bar with links for Home, Blog, Videos, Documentation, Community, Sponsor, a search icon, and a prominent orange "Download" button. Below the navigation bar, the title "Zed Attack Proxy (ZAP)" is displayed in a large, bold, dark font. A descriptive paragraph follows, stating: "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 shield-shaped character with a lightning bolt on its chest, a smiling face, and arms holding a sword. Below the main content area are two buttons: a blue "Quick Start Guide" button and an orange "Download Now" button.

Zed Attack Proxy (ZAP)

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.

Quick Start Guide   Download Now

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



# OWASP ZAP

## Automated Scan to target url

<

### Automated Scan



This screen allows you to launch an automated scan against an application – just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack:

Use traditional spider:

Use ajax spider:  with

Progress: Not started

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



Untitled Session - ZAP 2.14.0

Standard Mode < Quick Start Request Response Requester +

Sites Default Context Sites https://demo.owasp-juice.shop GET:assets assets GET:ftp GET:main.js GET:polyfills.js GET:robots.txt GET:runtime.js GET:sitemap.xml GET:styles.css GET:vendor.js

# Automated Scan

This screen allows you to launch an automated scan against an application – just enter its URL below and press 'Attack'. Please be aware that you should only attack applications that you have been specifically given permission to test.

URL to attack:  Select...

Use traditional spider:

Use ajax spider:  with Chrome Headless

Progress: Actively scanning (attacking) the URLs discovered by the spider(s)

History Search Alerts Output Spider Active Scan +

New Scan Progress: 0: https://demo.owasp-juice.shop 93% Current Scans: 1 Num Requests: 244 New Alerts: 0 Export

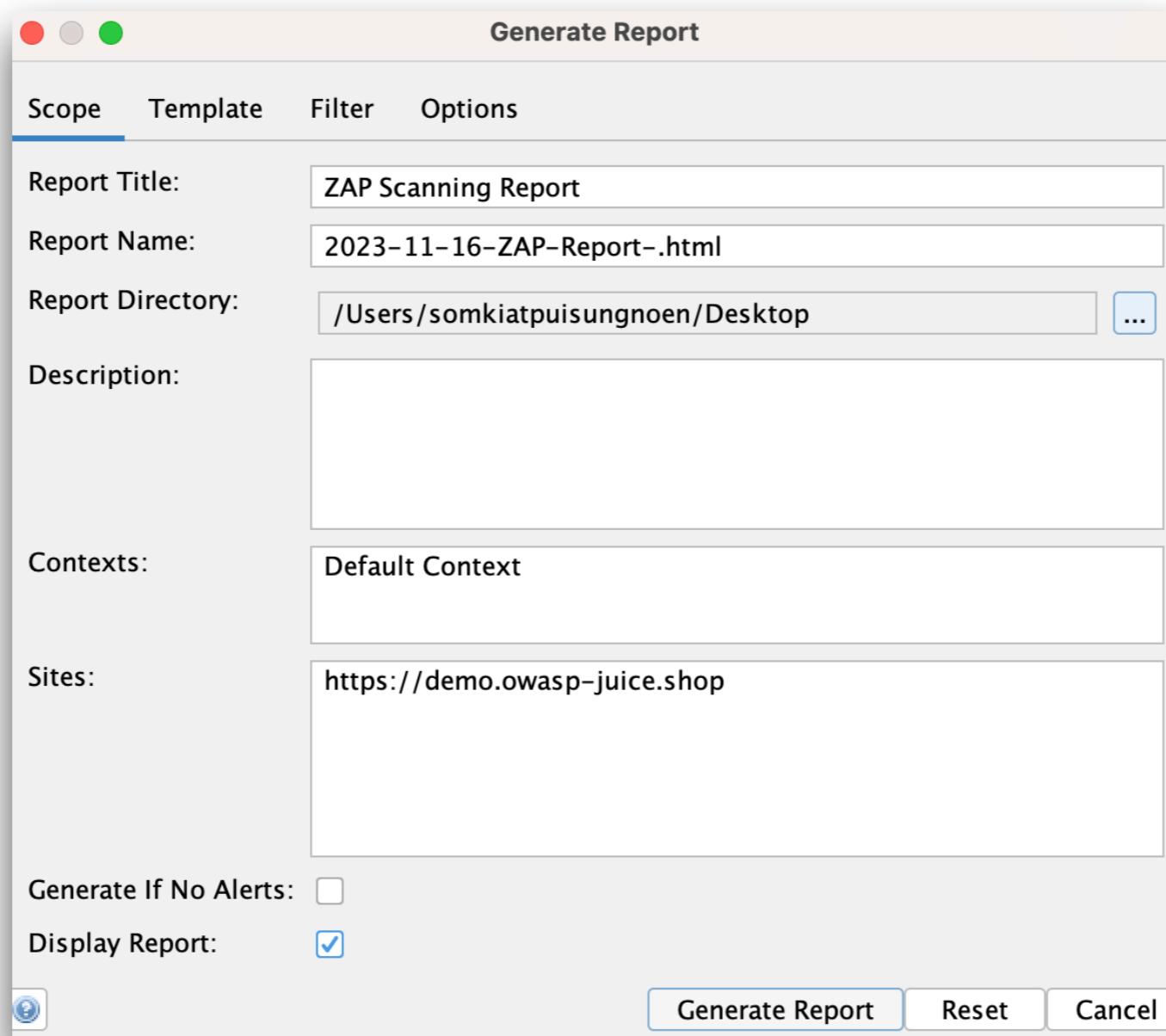
Sent Messages Filtered Messages

ID	Req. Timestamp	Resp. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp. Body
412	11/16/23, 10:48:22 PM	11/16/23, 10:48:22 PM	GET	https://demo.owasp-juice.shop/ftp	200	OK	1.19 s	709 bytes	506 bytes
413	11/16/23, 10:48:25 PM	11/16/23, 10:48:26 PM	GET	https://demo.owasp-juice.shop/main.js	200	OK	1.44 s	985 bytes	490,757 bytes
414	11/16/23, 10:48:25 PM	11/16/23, 10:48:27 PM	GET	https://demo.owasp-juice.shop/styles.css	200	OK	2.08 s	971 bytes	629,523 bytes
415	11/16/23, 10:48:26 PM	11/16/23, 10:48:28 PM	GET	https://demo.owasp-juice.shop/main.js	200	OK	1.69 s	977 bytes	490,757 bytes
416	11/16/23, 10:48:27 PM	11/16/23, 10:48:29 PM	GET	https://demo.owasp-juice.shop/styles.css	200	OK	2.1 s	963 bytes	629,523 bytes
417	11/16/23, 10:48:26 PM	11/16/23, 10:48:29 PM	GET	https://demo.owasp-juice.shop/ftp	503	Service Unav...	3.62 s	709 bytes	506 bytes
418	11/16/23, 10:48:19 PM	11/16/23, 10:48:30 PM	GET	https://demo.owasp-juice.shop/vendor.js	200	OK	11.08 s	975 bytes	1,421,052 bytes
419	11/16/23, 10:48:29 PM	11/16/23, 10:48:31 PM	GET	https://demo.owasp-juice.shop/styles.css	200	OK	1.58 s	959 bytes	629,523 bytes
420	11/16/23, 10:48:31 PM	11/16/23, 10:48:32 PM	GET	https://demo.owasp-juice.shop/styles.css	200	OK	1.49 s	967 bytes	629,523 bytes
421	11/16/23, 10:48:29 PM	11/16/23, 10:48:33 PM	GET	https://demo.owasp-juice.shop/ftp	503	Service Unav...	3.72 s	709 bytes	506 bytes
422	11/16/23, 10:48:32 PM	11/16/23, 10:48:34 PM	GET	https://demo.owasp-juice.shop/styles.css	200	OK	1.57 s	959 bytes	629,523 bytes
423	11/16/23, 10:48:34 PM	11/16/23, 10:48:35 PM	GET	https://demo.owasp-juice.shop/styles.css	200	OK	1.56 s	959 bytes	629,523 bytes
424	11/16/23, 10:48:33 PM	11/16/23, 10:48:37 PM	GET	https://demo.owasp-juice.shop/ftp	503	Service Unav...	3.47 s	709 bytes	506 bytes
425	11/16/23, 10:48:30 PM	11/16/23, 10:48:40 PM	GET	https://demo.owasp-juice.shop/vendor.js	200	OK	9.15 s	983 bytes	1,421,052 bytes
426	11/16/23, 10:48:37 PM	11/16/23, 10:48:40 PM	GET	https://demo.owasp-juice.shop/ftp	503	Service Unav...	3.62 s	709 bytes	506 bytes
427	11/16/23, 10:48:40 PM	11/16/23, 10:48:44 PM	GET	https://demo.owasp-juice.shop/ftp	503	Service Unav...	3.56 s	713 bytes	506 bytes
428	11/16/23, 10:48:40 PM	11/16/23, 10:48:46 PM	GET	https://demo.owasp-juice.shop/vendor.js	200	OK	6.54 s	975 bytes	1,421,052 bytes
429	11/16/23, 10:48:44 PM	11/16/23, 10:48:48 PM	GET	https://demo.owasp-juice.shop/ftp	503	Service Unav...	3.76 s	709 bytes	506 bytes



# Generate Report

Report -> Generate Report



# HTML Report

## ZAP Scanning Report

Generated with  ZAP on Thu 16 Nov 2023, at 22:51:06

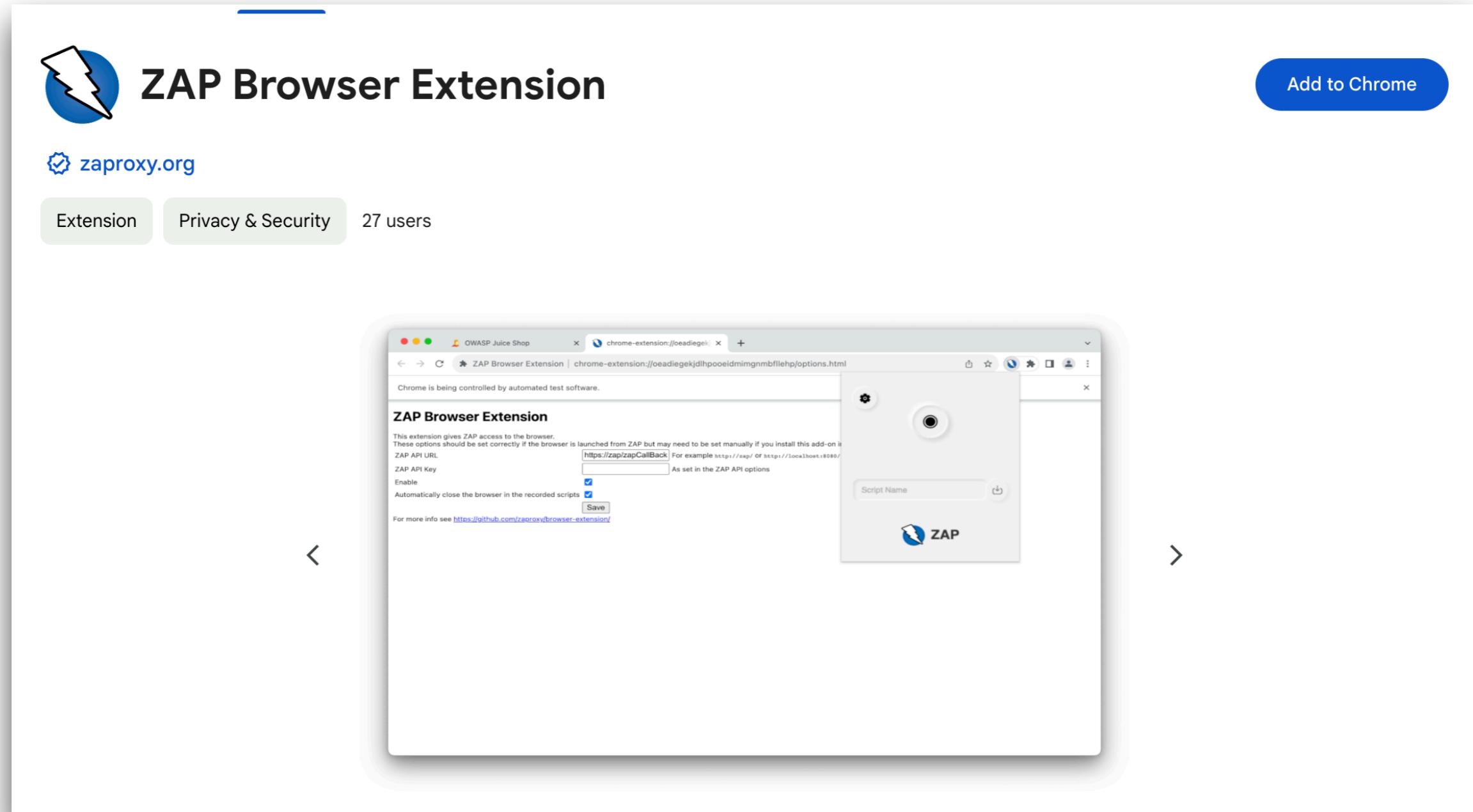
ZAP Version: 2.14.0

## Contents

- [About this report](#)
  - [Report parameters](#)
- [Summaries](#)
  - [Alert counts by risk and confidence](#)
  - [Alert counts by site and risk](#)
  - [Alert counts by alert type](#)
- [Alerts](#)
  - [Risk=Medium, Confidence=High \(1\)](#)



# ZAP on Google Chrome



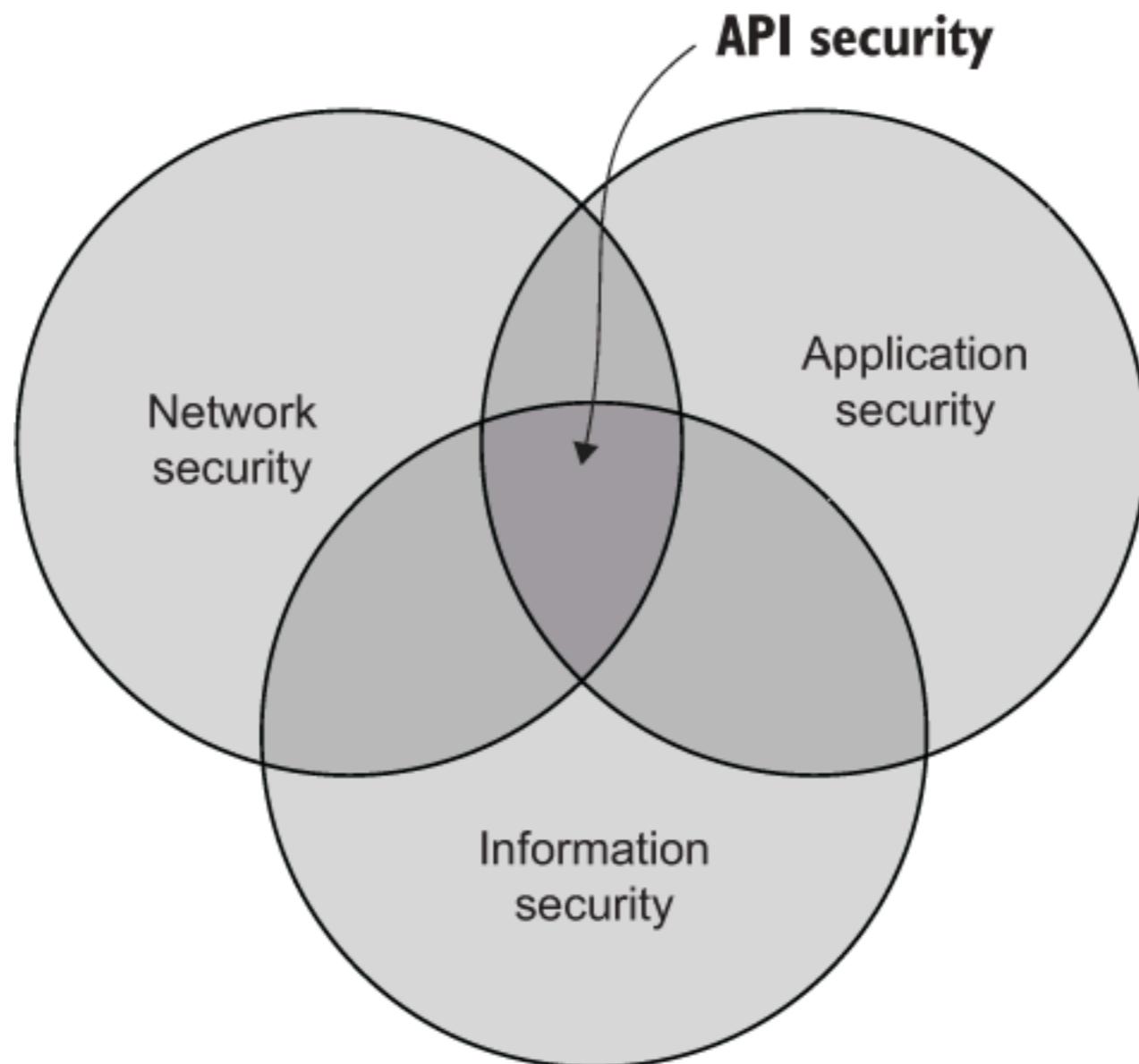
<https://chromewebstore.google.com/detail/zap-browser-extension/cnmficmodhagepcmhogkbdakncebckho?pli=1>



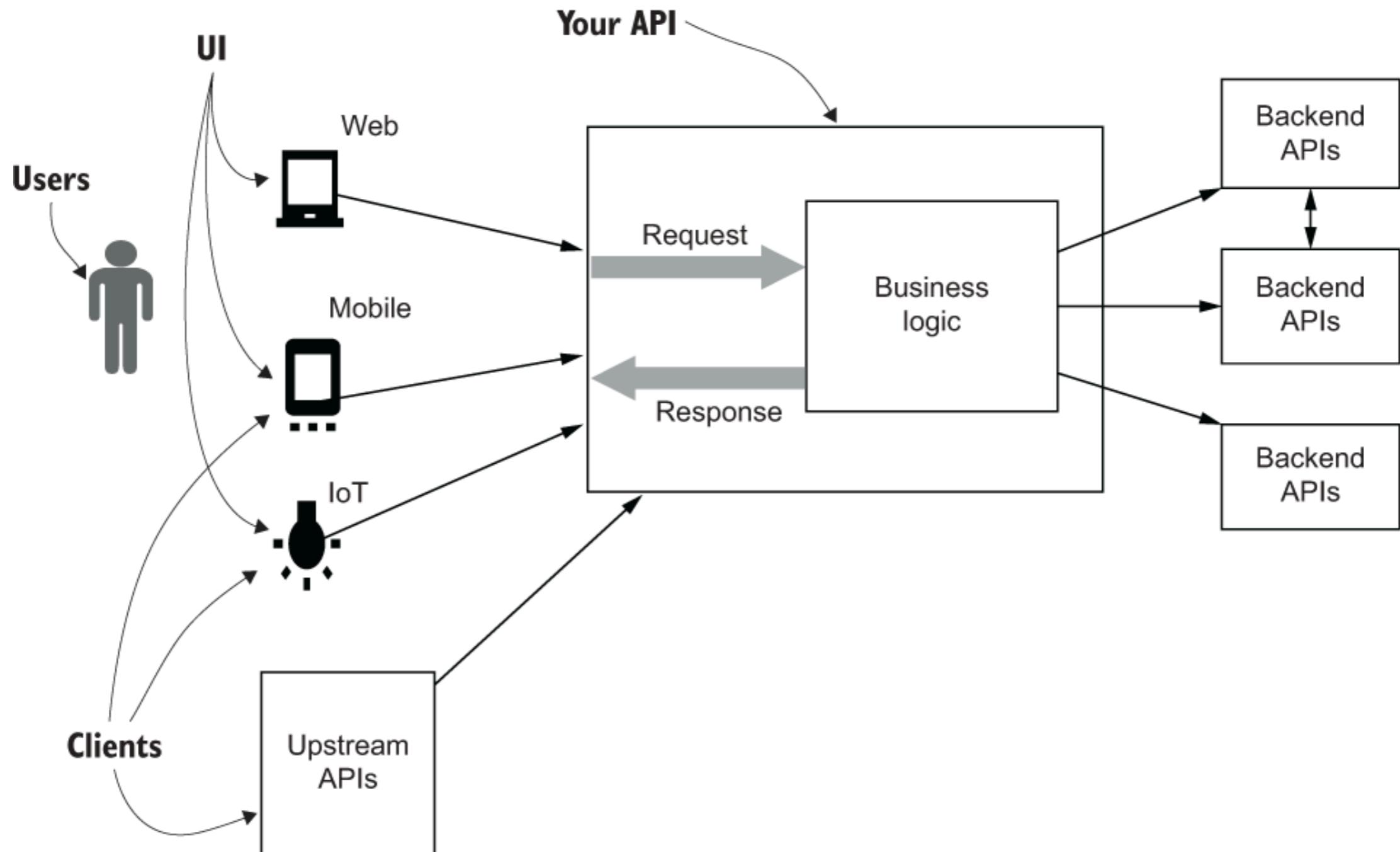
# Architecture

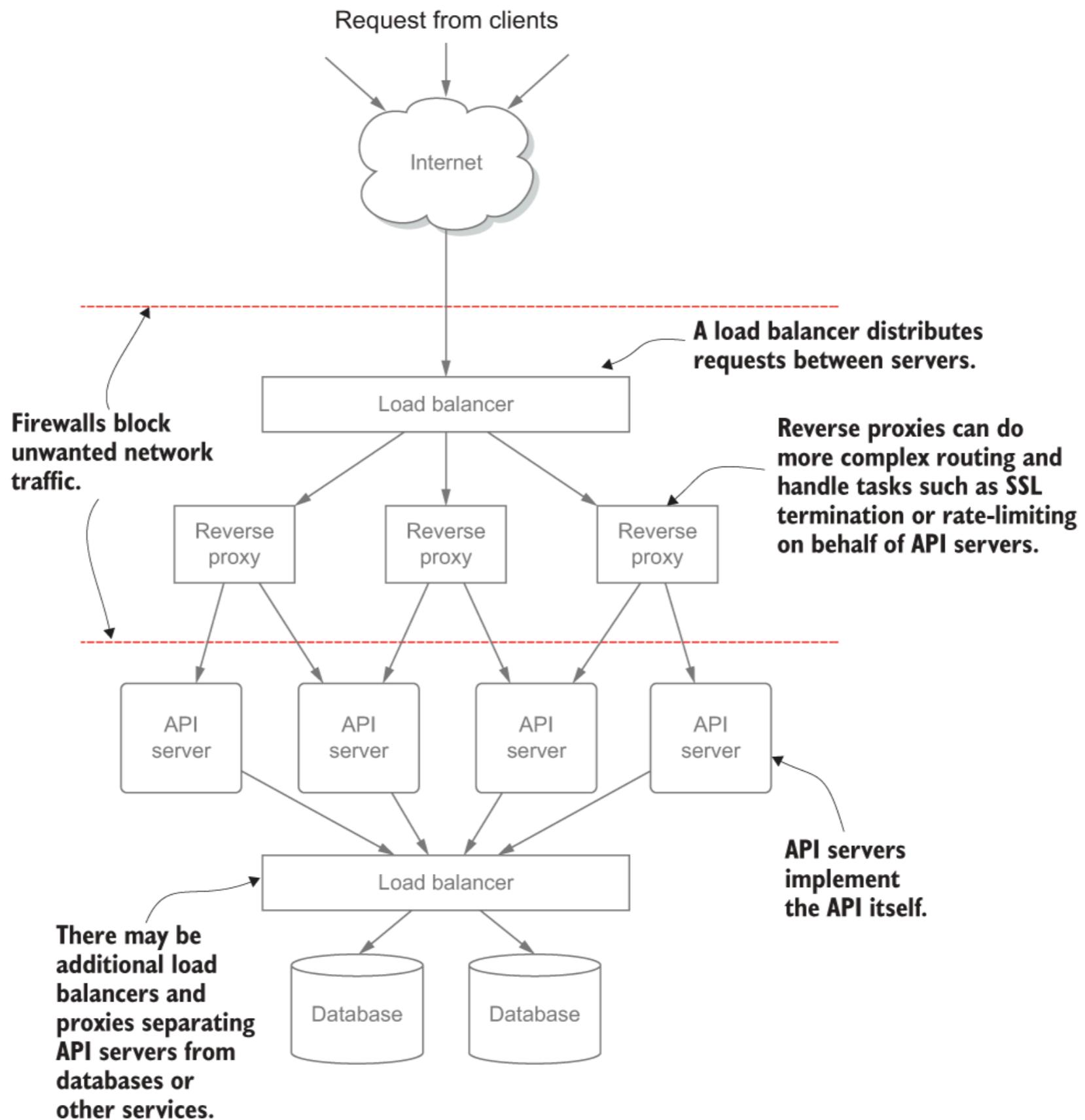


# API Security



# Simple Architecture





# Security Mechanisms

Authorization

Authentication

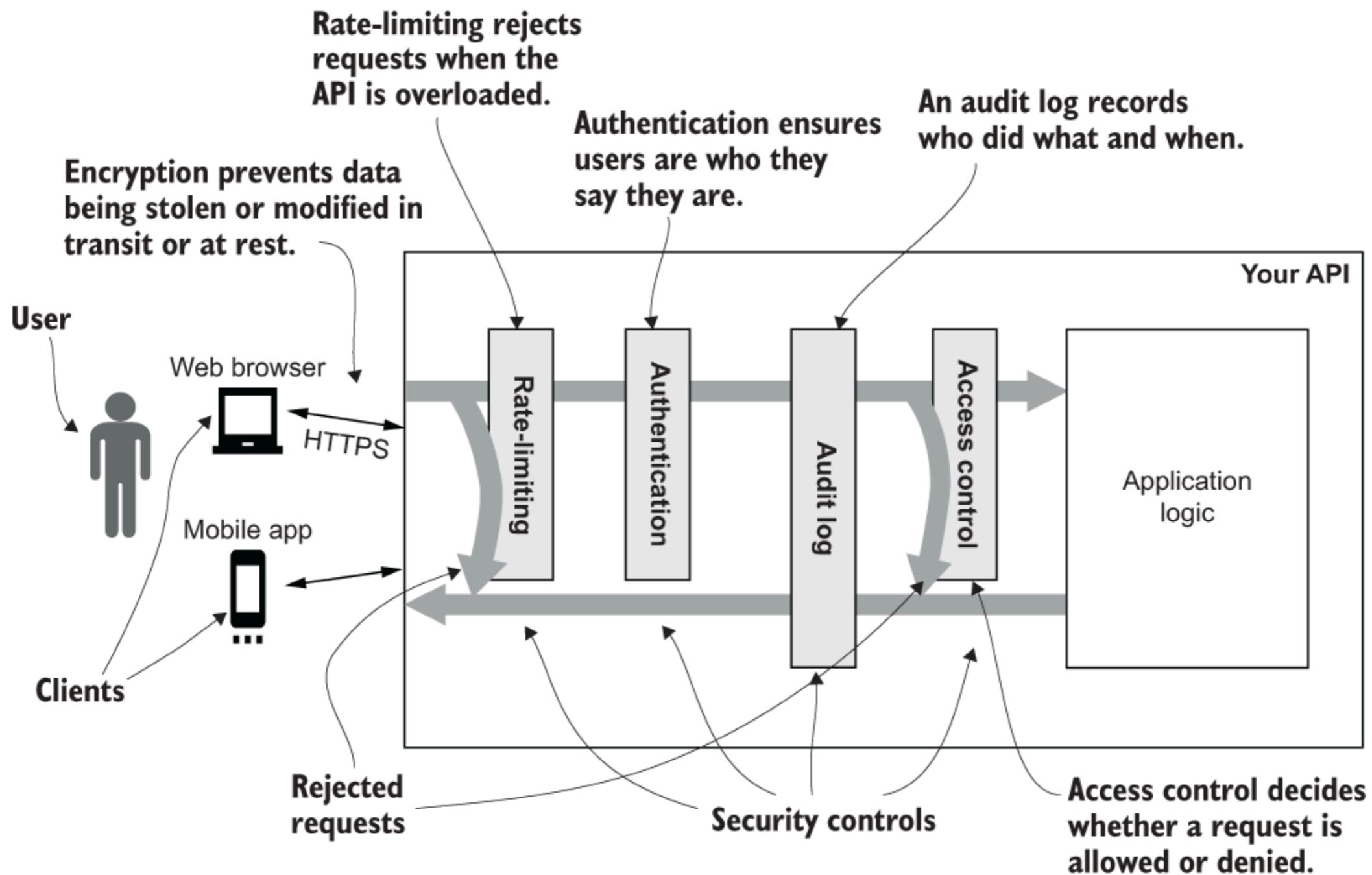
Encryption

Audit logging

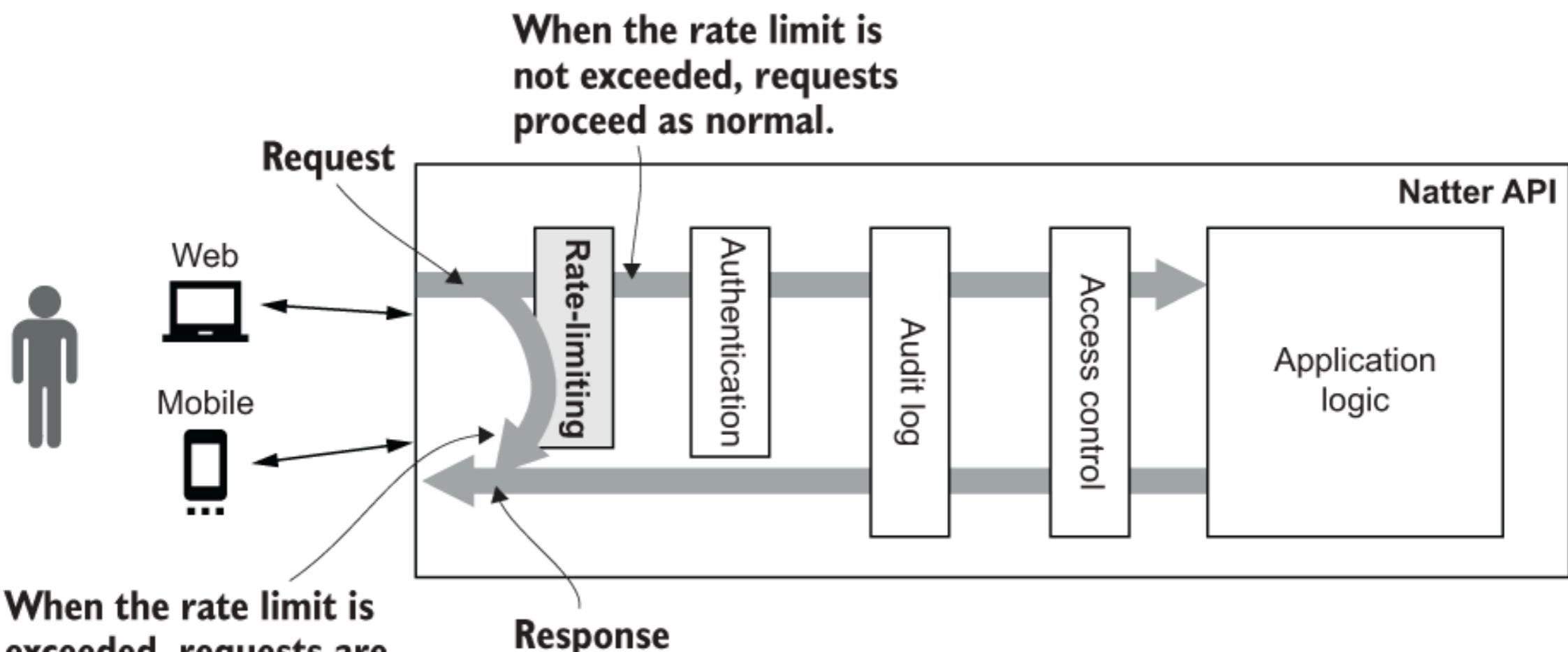
Rate limiting



# Security Mechanisms



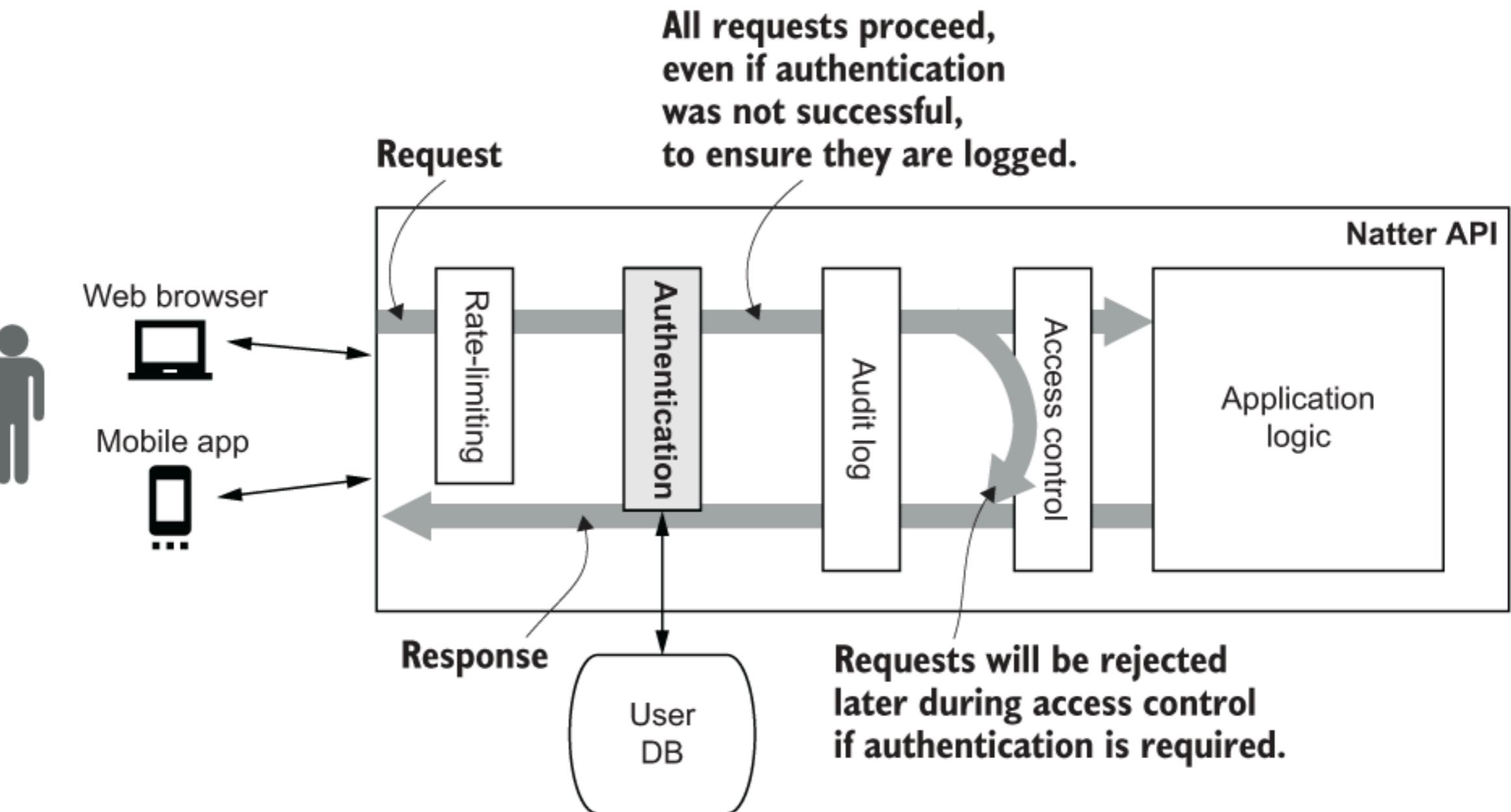
# Rate Limiting for Availability



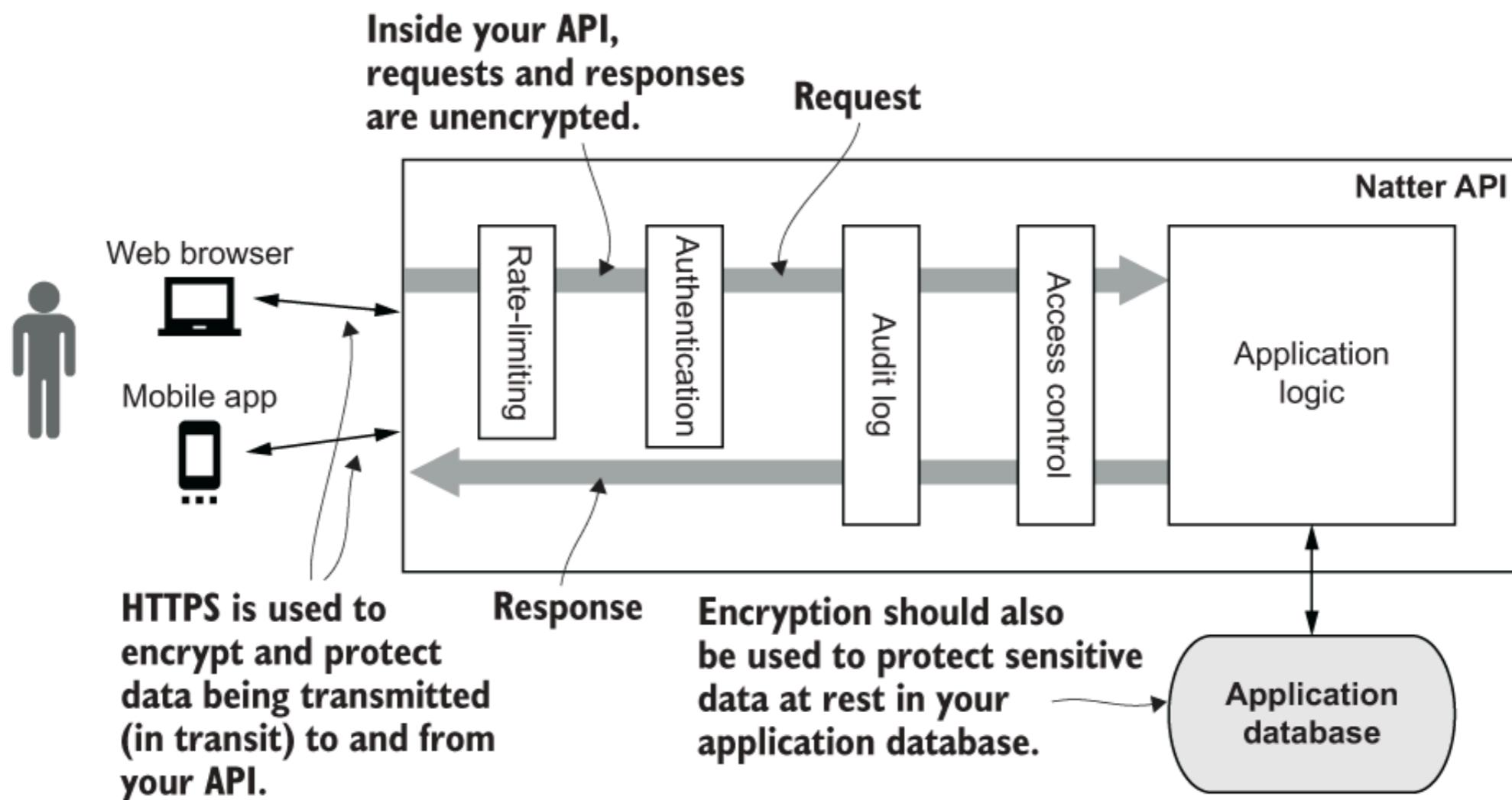
When the rate limit is exceeded, requests are immediately rejected with a 429 Too Many Requests HTTP status code.



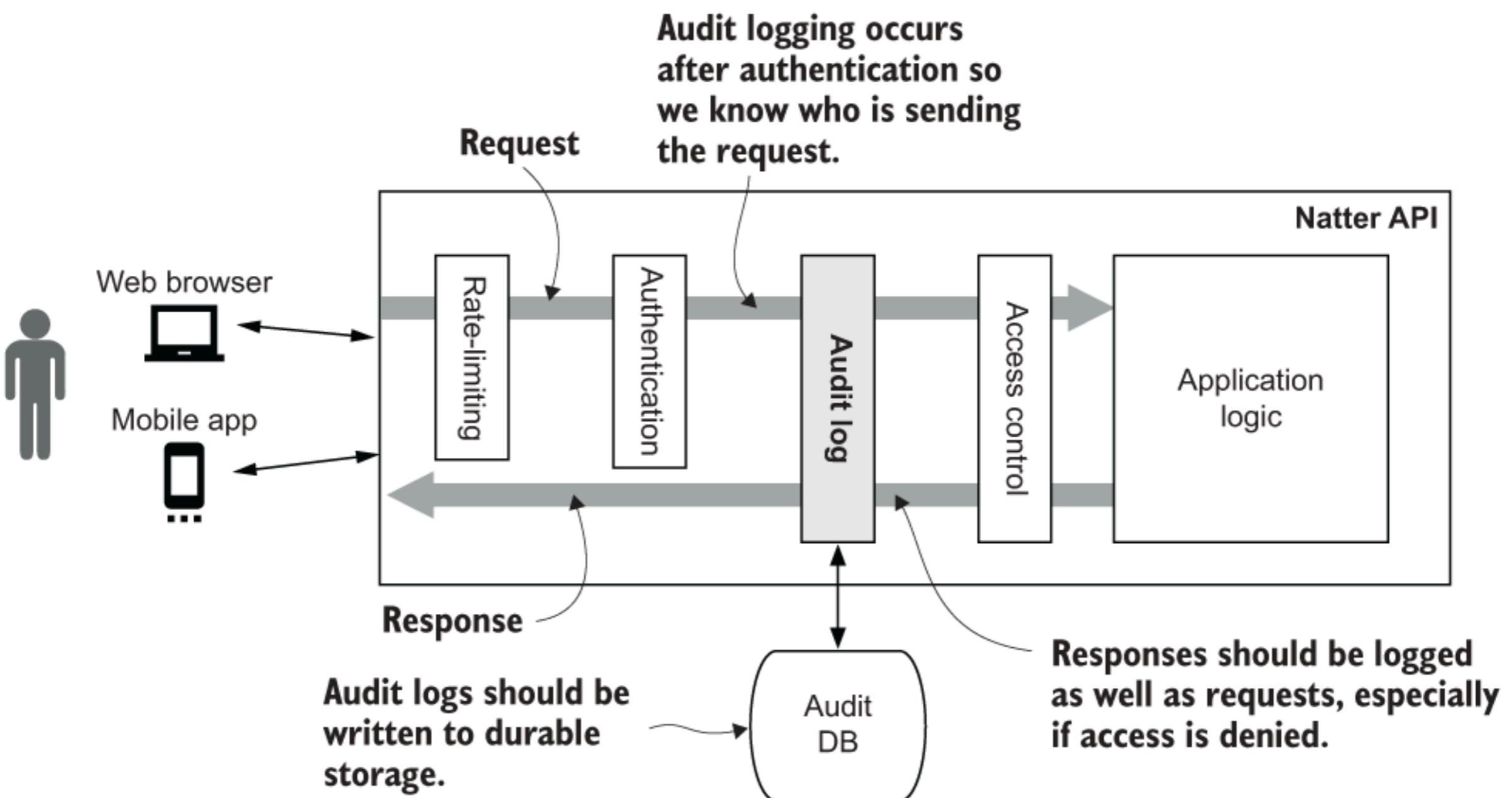
# Authentication



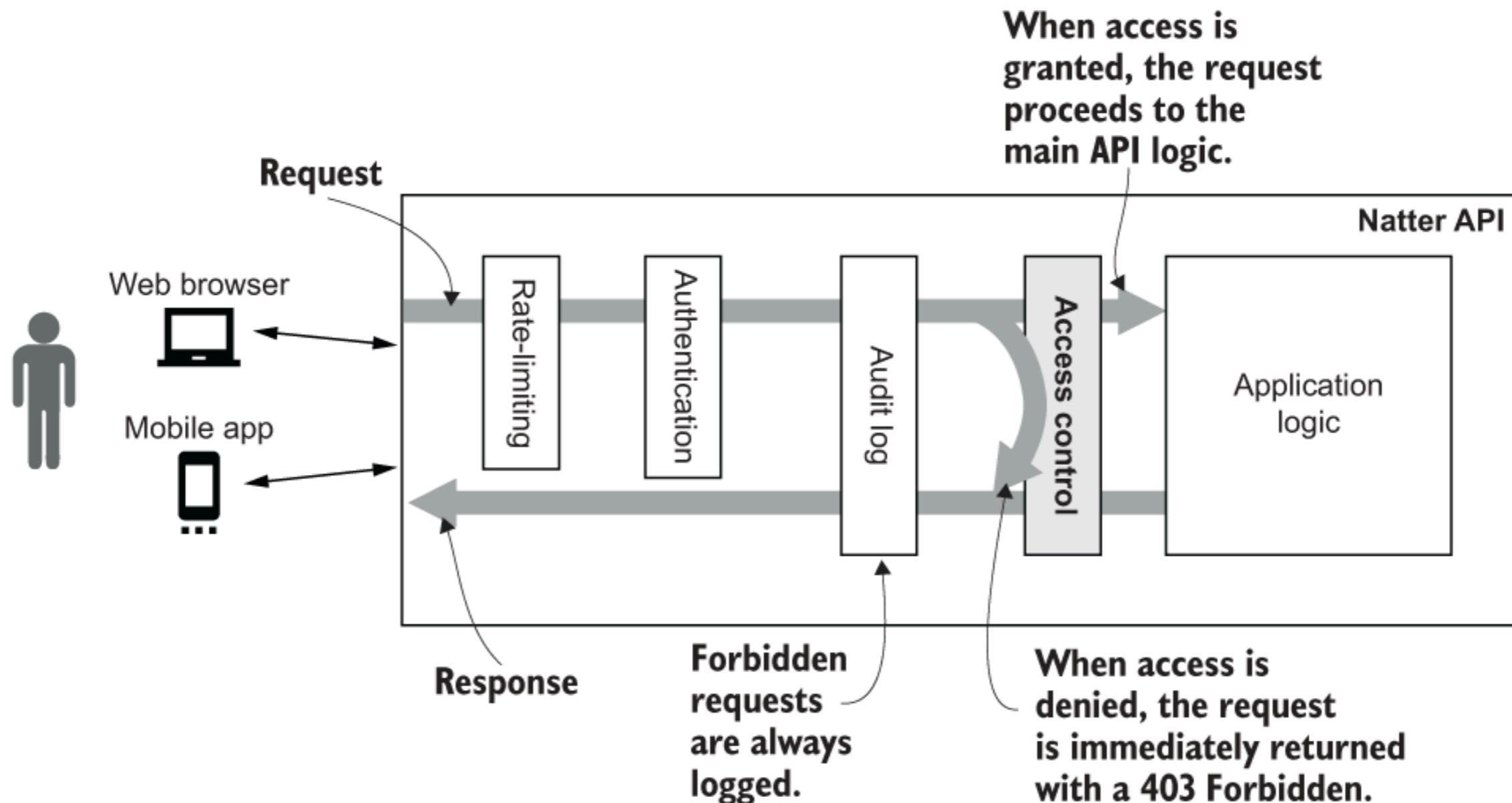
# Use encryption to keep data private



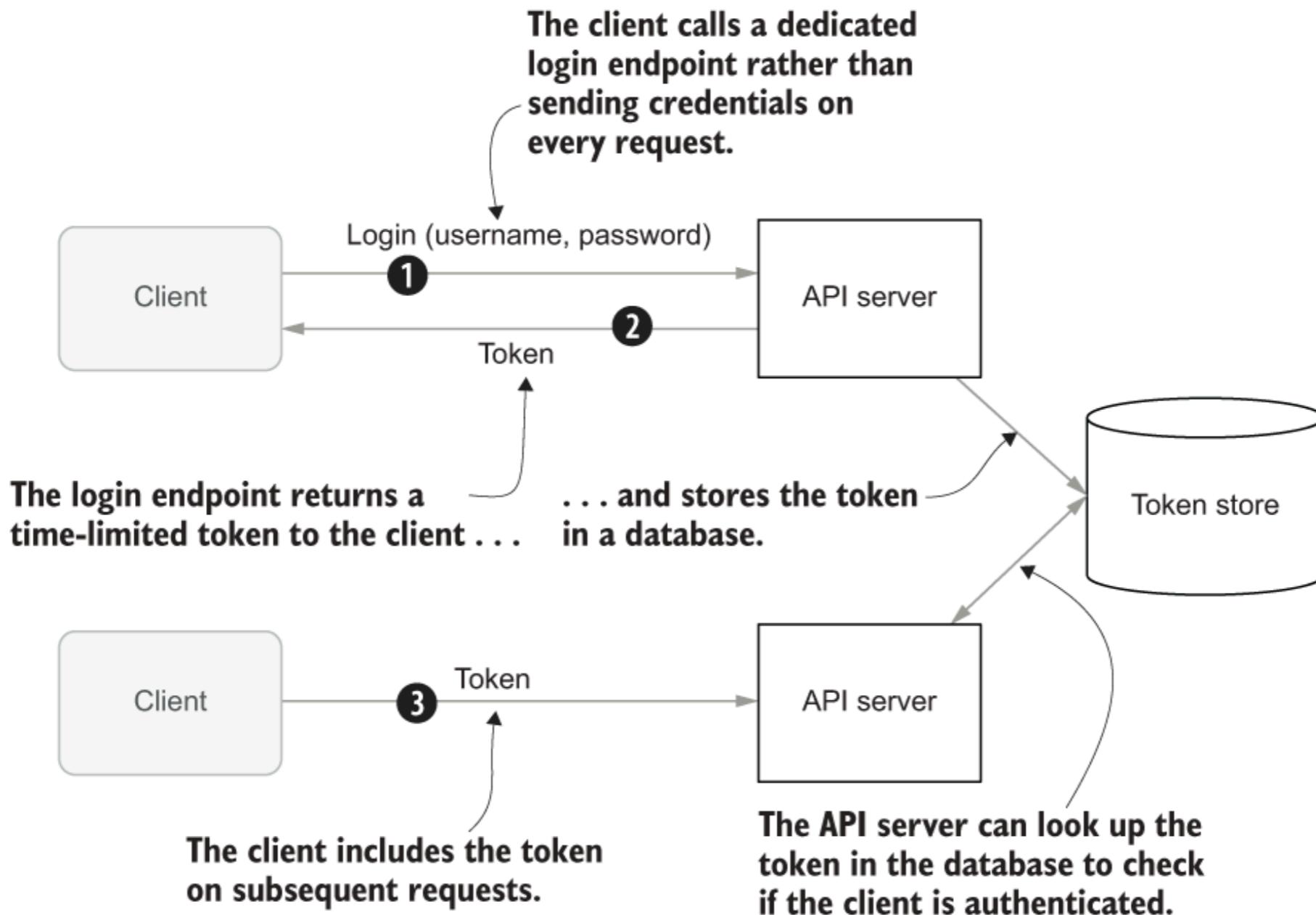
# Audit logging for accountability



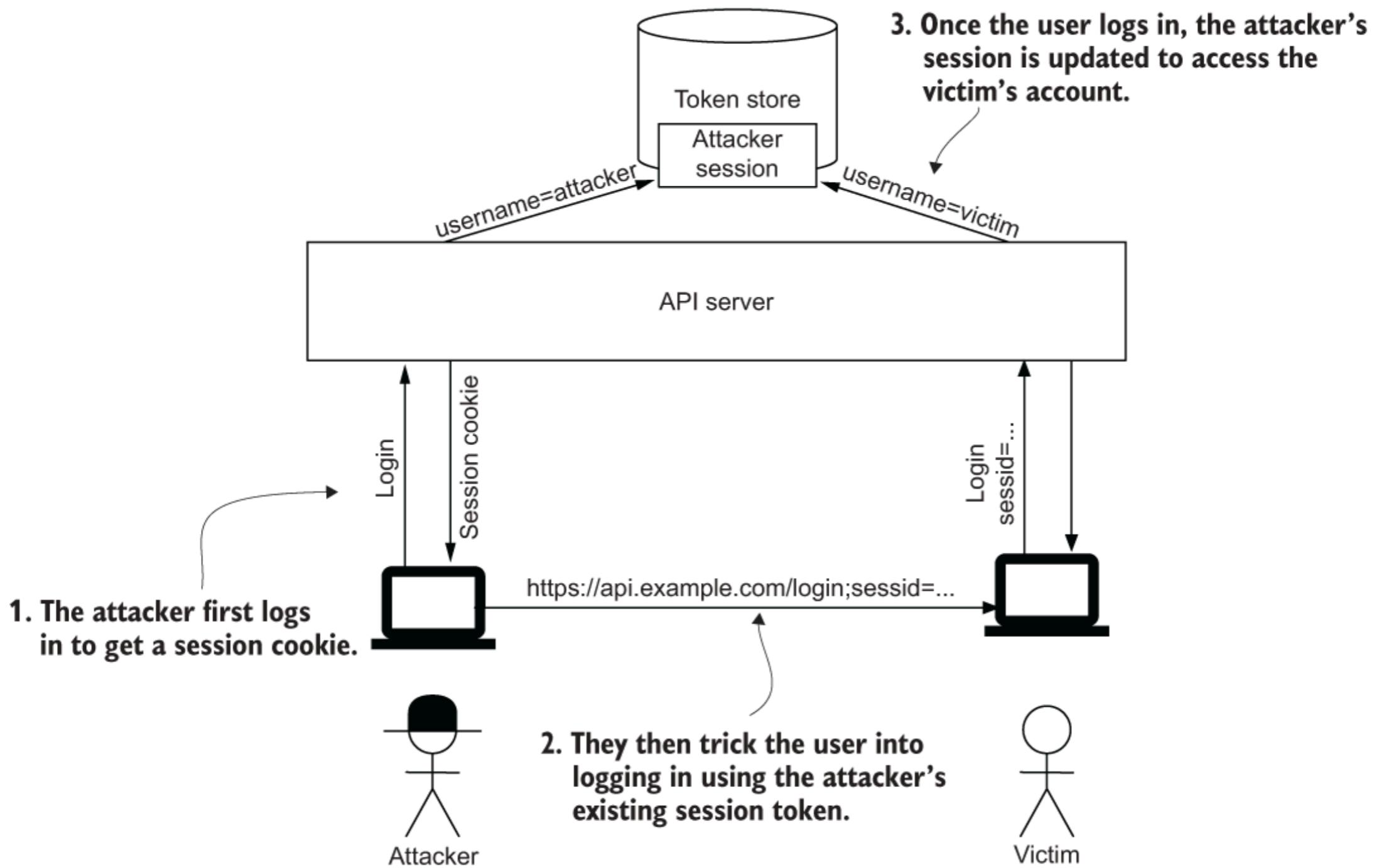
# Authorization or Access control



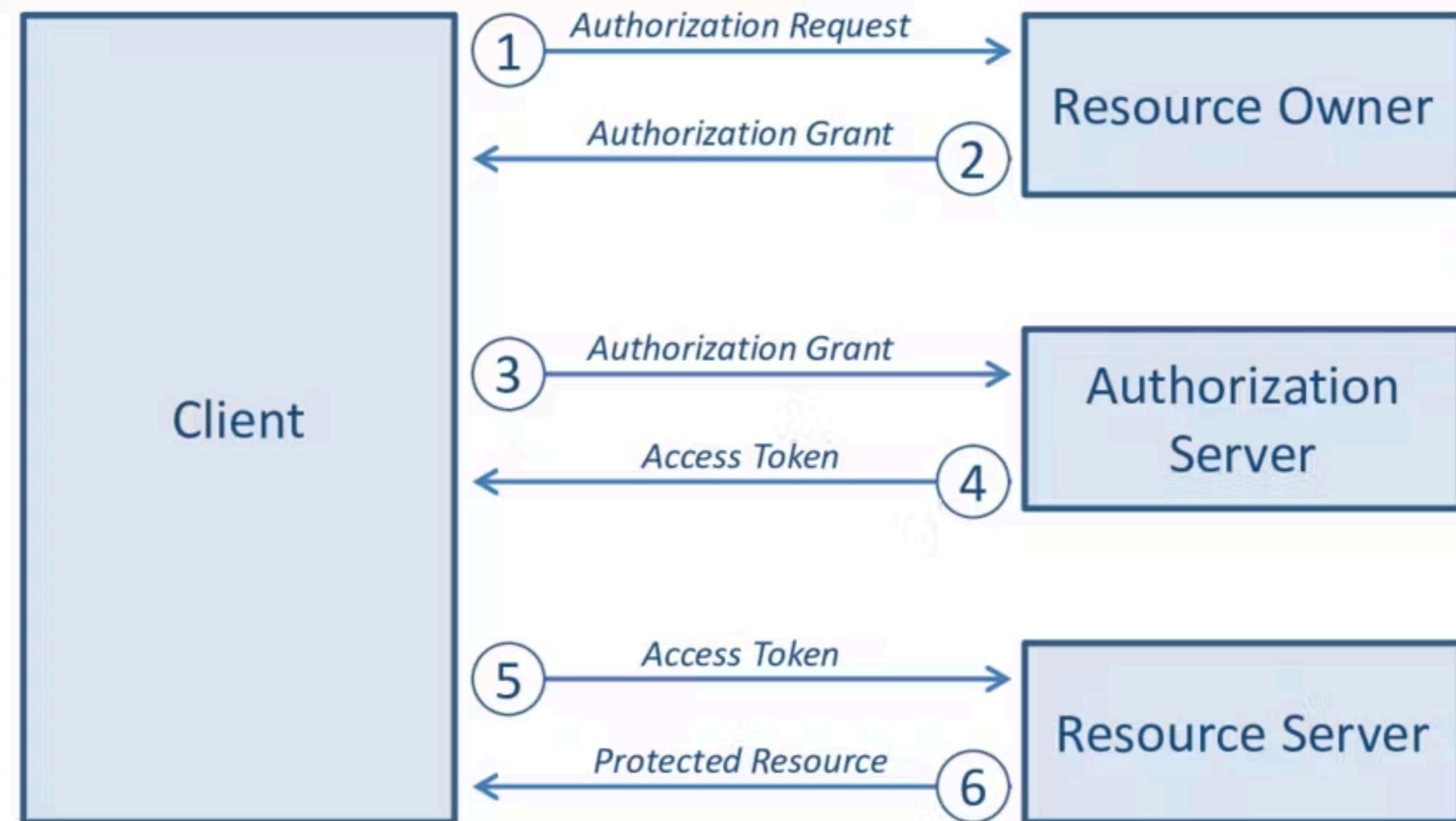
# Basic of Token-based



# Attack !!

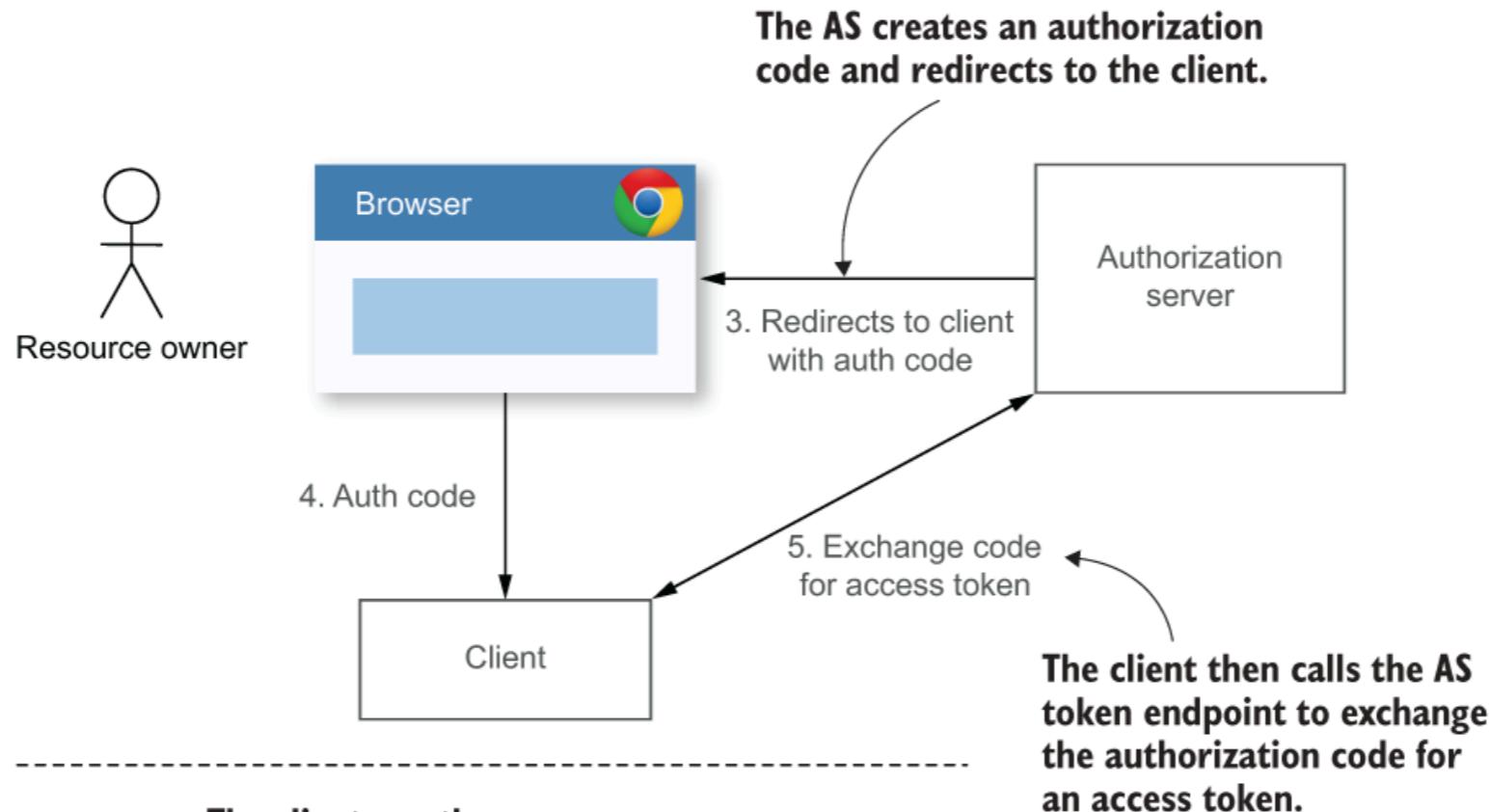
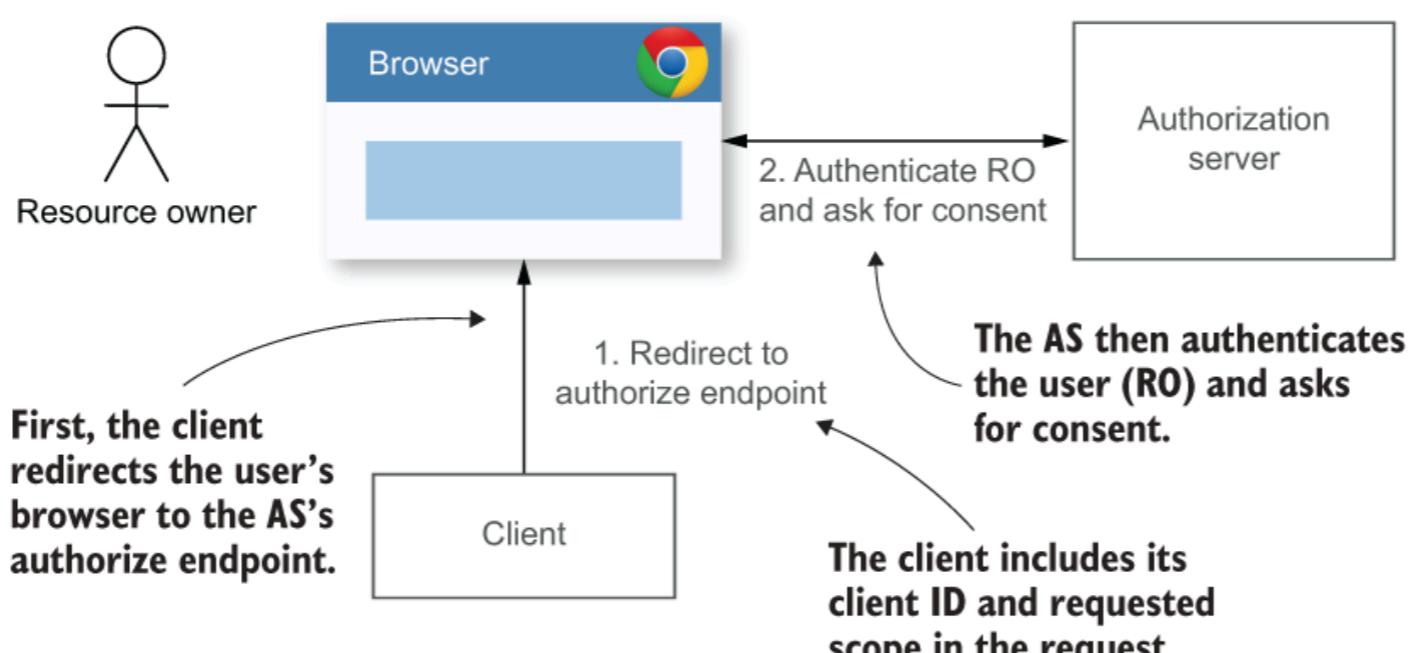


# OAuth 2.0



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





**The client can then use the access token to access the API.**



# Q/A

