CODE REVIEW: SECURITY

What is about to be worked ?

- Introduction
- Code Review to check the software's security quality
- Concrete case

Introduction

What?

- Reread
- Norm
- Simplicity
- Clarity

Why?

- -33% maintenance
- -60% bugs (only 25% for unit test)
- Maintenance
- Reliability
- Evolutivity

When? How?

- Before push
- Before Merge
- Continuous
- Piece by piece
- Positive hints
- Fix by developer
- Formation

Introduction

CODE REVIEW: SECURITY

Usefulness ?

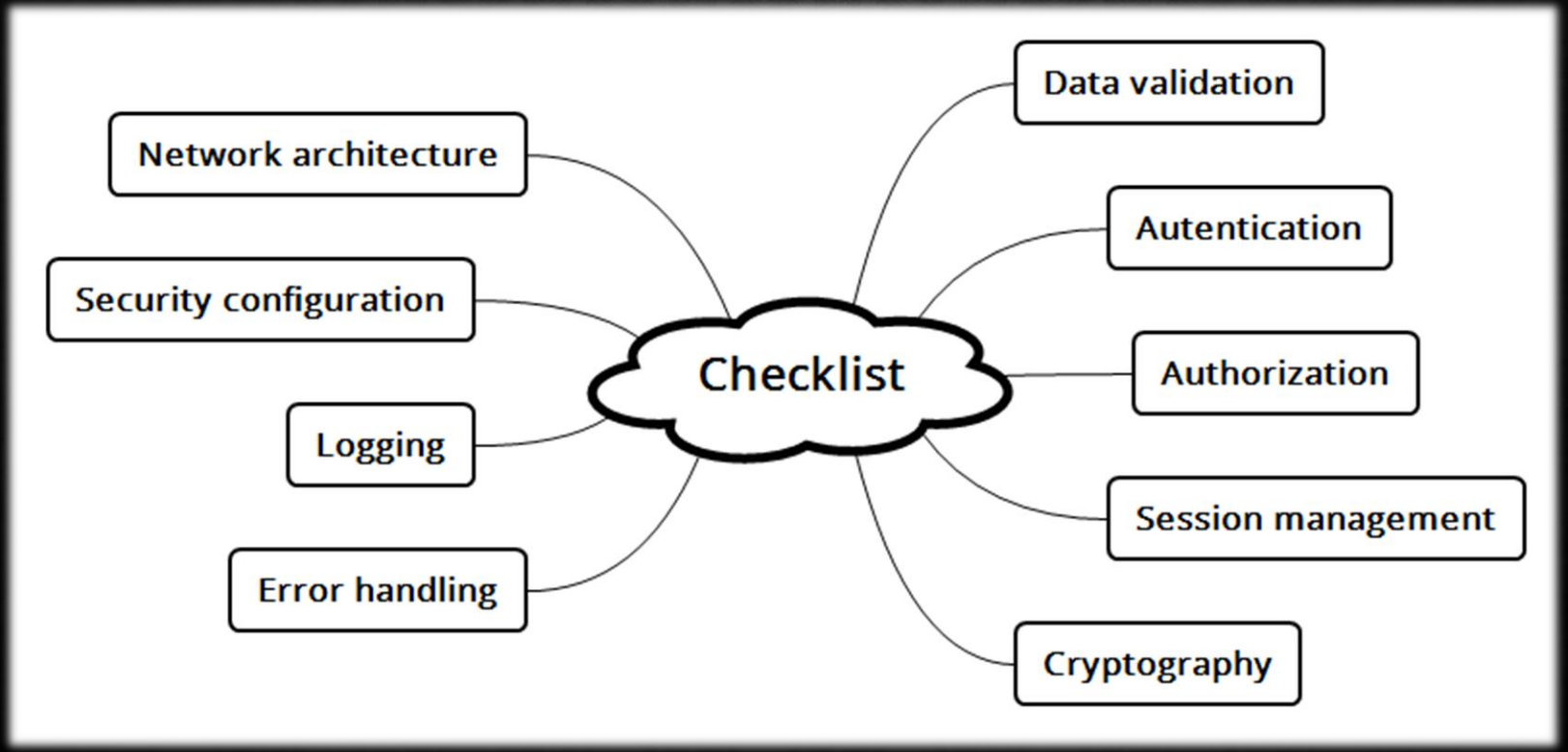
Identify bugs

Fix bugs

Before it becomes a security vulnerability

How ?

More criticals securities area



Example with the Authorization checklist

STRIDE Threat List		
Type	**Examples**	**Security Control**
Spoofing	Threat action aimed to illegally access and use another user's credentials, such as username and password	Authentication
Tampering	Threat action aimed to maliciously change/modify persistent data, such as persistent data in a database, and the alteration of data in transit between two computers over an open network, such as the Internet	Integrity
Repudiation	Threat action aimed to perform illegal operation in a system that lacks the ability to trace the prohibited operations.	Non-Repudiation
Information disclosure.	Threat action to read a file that they were not granted access to, or to read data in transit.	Confidentiality
Denial of service.	Threat aimed to deny access to valid users such as by making a web server temporarily unavailable or unusable.	Availability
Elevation of privilege.	Threat aimed to gain privileged access to resources for gaining unauthorized access to information or to compromise a system.	Authorization

Example with the Authorization checklist

Threat Category	Affects Processes	Affects Data Stores	Affects External Entities	Affects Data Flows
Spoofing	Y		Y	
Tampering	Y	Y		Y
Repudiation		Y	Y	Y
Information Disclosure	Y	Y		Y
Denial of Service	Y	Y		Y
Elevation of Privilege	Y			

CONCRETE CASE

Concrete case: Data validation

Data Protection in Storage and Transit	1. Standard encryption algorithms and correct key sizes are being used 2. Hashed message authentication codes (HMACs) are used to protect data integrity 3. Secrets (e.g. keys, confidential data) are cryptographically protected both in transport and in storage 4. Built-in secure storage is used for protecting keys 5. No credentials and sensitive data are sent in clear text over the wire
Data Validation / Parameter Validation	1. Data type, format, length, and range checks are enforced 2. All data sent from the client is validated 3. No security decision is based upon parameters (e.g. URL parameters) that can be manipulated 4. Input filtering via white list validation is used 5. Output encoding is used

Concrete case: Data validation

Reviewing Code for Data Validation

http://www.owasp.org/index.php/Reviewing_Code_for_Data_Validation

```
<?php

// On récupère les variables envoyées par le formulaire
$login = $_POST['login'];
$password = $_POST['password'];

// Requête SQL
$req = $bdd->query("SELECT * FROM utilisateurs WHERE login='$login' AND
password='$password'");
```

Pas de data processing

Pas de Hashage

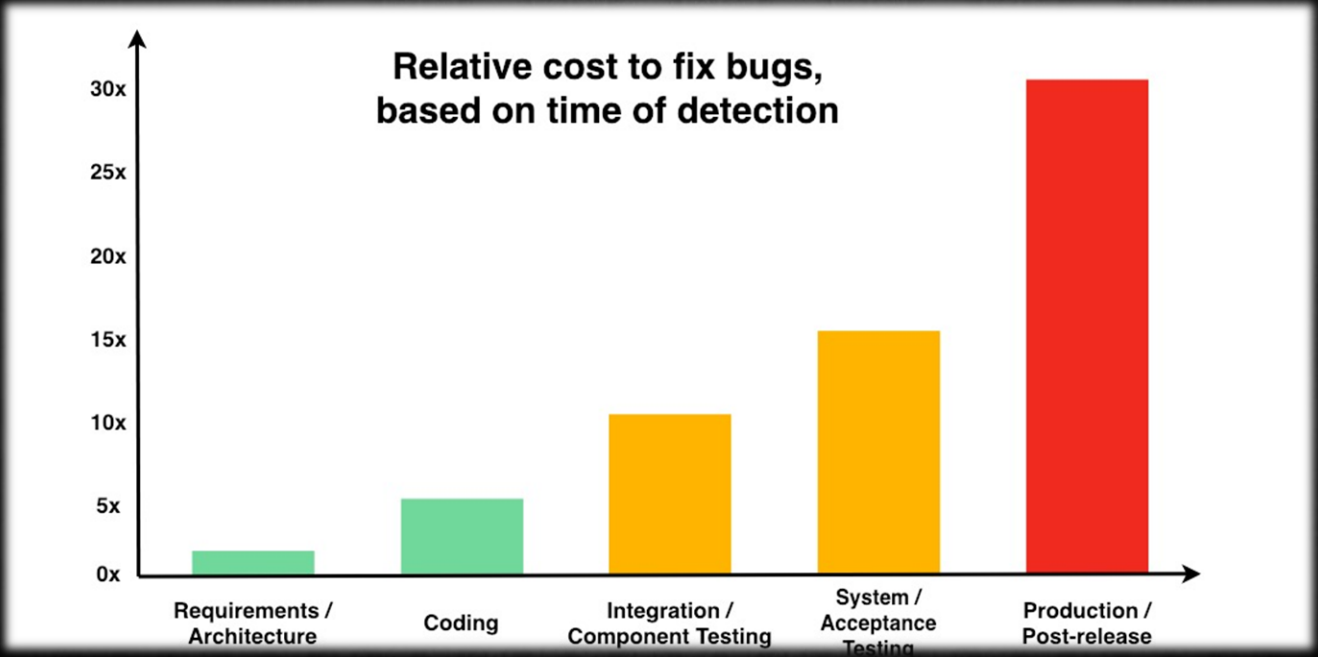
Dans le cas ou `$login = "jean' #"`

```
$req = $bdd->query("SELECT * FROM utilisateurs WHERE login='jean' # AND
password="");

// Qui sera interprété de la façon suivante

$req = $bdd->query("SELECT * FROM utilisateurs WHERE login='jean'");
```

Finally...



Sources:
- https://www.owasp.org/images/2/2e/OWASP_Code_Review_Guide-V1_1.pdf
- https://openclassrooms.com/fr/courses/2091901-protegez-vous-efficacement-contre-les-failles-web/2680180-linjection-sql
- https://deepsource.io/blog/exponential-cost-of-fixing-bugs/