Lesson 6

Build Safe and Secure Software

In this lesson we will talk about:

- Why modern software sucks
- Software Testing
- Static and Dynamic Analysis
- CI/CD Pipelines
- Testing is not enough

Year 2025

We are drowning in security vulnerabilities, software defects and bugs!

2025 Vulnerability Stats

JSON

Click here to view 2024 Stats

It's May 10 and 17,473 vulnerabilities have been published in 2025.

This marks an **increase of 23%** compared to this time last year.

17,473

Vulnerabilities published in 2025

134

Avg. new vulnerabilities each day!

23%

Increase in vulnerabilities YoY

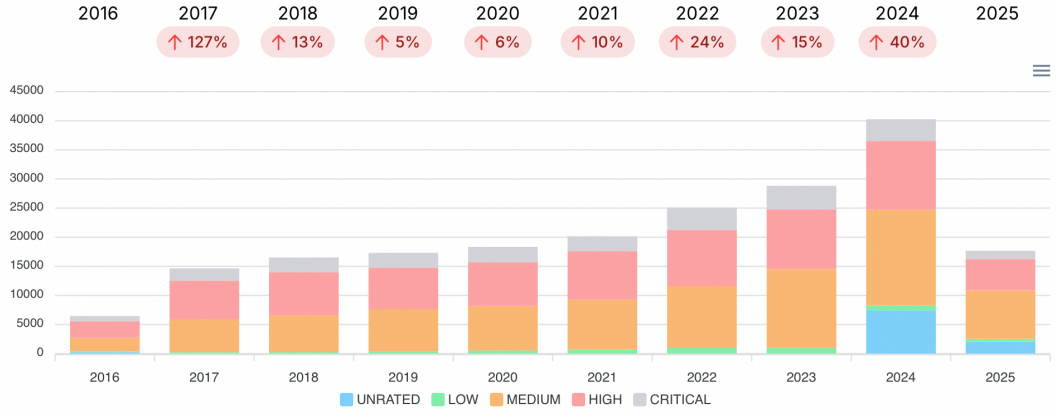
More than 100 security vulnerabilities every single day!

May 2025

23%
increase compared to this time last year!

Source: securityvulnerability.io/stats

2016 - 2025 Vulnerability Severity By Year



But we have SAFe, different agile methodologies, cloud computing and powerful computers, still cant produce safe and secure software

Why?

Some might say: we dont test enough or good enough. Some others, that we have developed too quickly, too much.

The truth?

GET READY

SAFE AND SECURE PROGRAMMING

What does it mean to build safe and secure software?

No advanced maths, no expensive software, no extra licenses required

Safe Software

"Software safety is an engineering discipline that aims to ensure that software, which is used in critical related systems does not contribute to any hazards such a system might pose."

- Automotive (ISO 26262), railway software (EN 50716)
- Airborne software (DO-178C/ED-12C)
- Air traffic management software (DO-278A/ED-109A)
- Medical devices (IEC 62304)
- Nuclear power plants (IEC 60880)

System Safety

"System Safety aims to achieve safety by reducing risks in technical systems to an acceptable level."

System Safety

"Functional safety is achieved through engineering development to ensure correct execution and behavior of software functions as intended"

System Safety

"Safety consistent with mission requirements, is designed into the software in a timely, cost effective manner."

System Safety and AI

"Software that employs artificial intelligence techniques such as machine learning follows a radically different lifecycle. Currently, standards generally do not endorse their use. For example, EN 50716 states that artificial intelligence and machine learning are not recommended for any safety integrity level."

Secure Software

"Security is resilience against harm caused by others."

- A threat
- DDOS atack
- Cryptojacking
- Man In The Middle (MITM) attacks

Secure Software

"Secure coding is the practice of developing computer software in such a way that guards against the accidental introduction of security vulnerabilities. Defects, bugs and logic flaws are consistently the primary cause of commonly exploited software vulnerabilities."

Secure Software

- Buffer-overflow prevention
- Format-string attack prevention
- Integer-overflow prevention
- Path traversal prevention

Many IT organizations are currently working hard to adapt secure software development life cycle for their products and services. Despite that the number of security vulnerabilities is increasing

Software Testing

Software Testing

Software testing is the act of checking whether our software satisfies certain expectations.

Software Testing

It is all about finding bugs. Bug = a defect in the code that causes an undesirable result. Bugs generally slow testing progress and involve programmer assistance to debug and fix.

Software Testing

But not all defects cause a failure. For example, a defect in dead code will not be considered a failure. Or a defect in the rendering the user manual documentation will not cause a general failure of the application.

So, we must know where and what to look.

Software Testing

Software testing can determine **the correctness** of software for specific scenarios but cannot determine correctness for **all** scenarios.

In other words, it cannot find **all** bugs.

Software Testing

So, by definition software testing already is telling us we cannot cover all possible situations to test. Despite that, many IT organizations believe testing alone can achieve that.

Is testing enough?

- Start testing your software
- Write different test cases
- Automate
- But testing will not be enough
- Go back to specifications

Software Testing

What can we do about it? Is it even possible to test all possible cases, or simple does not matter?

The correct answer:

It depends!

If you are building a complex application or algorithm, used in a business or mission critical installation

YES

You must think of **ALL** possible cases

In other words if your software is a business or mission critical application - **YES** - you must think how can you test your specs for ALL possible cases.

The more critical your algorithm, application, pipeline is, the more you must be aware how correctly it runs and it is implemented.

But how can you do that? And what does it really mean to test ALL possible cases?

Software Testing

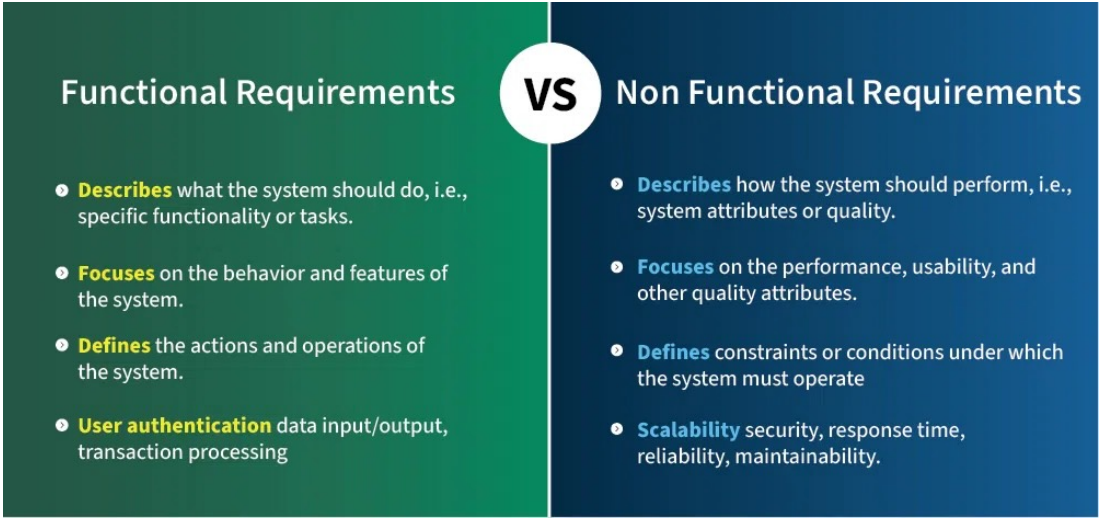
- Functional Testing
- Non-Functional Testing

Functional Testing

It is a software system testing that verifies whether a system or application meets its functional requirements or specifications.

Remember this?

A **functional software specification** is a written description of what our program is supposed to do.


Functional Requirements
VS
Non Functional Requirements
Describes what the system should do, i.e., specific functionality or tasks.
Focuses on the behavior and features of the system.
Defines the actions and operations of the system.
User authentication data input/output, transaction processing
Describes how the system should perform, i.e., system attributes or quality.
Focuses on the performance, usability, and other quality attributes.
Defines constraints or conditions under which the system must operate
Scalability security, response time, reliability, maintainability.

Functional Testing

- Unit Testing
- Integration System Testing
- Regression Testing
- Smoke Testing
- Usability Testing

Unit Testing

It is a component or module testing, where we are testing a part of our software only to ensure it is bug free.

- we test a very simple part
- can be a function or a method
- can be automated
- or manually executed

Unit testing is intended to ensure that the units meet their design and behave as intended.

Unit Testing

Unit testing finds problems early in the development cycle. This includes both bugs in the programmer's implementation and flaws or missing parts of the specification for the unit.

Unit Testing

The process of writing a thorough set of tests forces the author to think through inputs, outputs, and error conditions, and thus more crisply define the unit's desired behavior.

Non-functional Testing

Non-functional testing is testing software for its non-functional requirements: the way a system or application operates: does it use a lot of computing resources ...

Functional Testing

- Endurance Testing
- Performance Testing
- Scalability Testing
- Stress Testing

Static and Dynamic Analysis

Static vs Dynamic Analysis

Static program analysis is the analysis of computer programs performed without executing them, in contrast with dynamic program analysis, which is performed on programs during their execution in the integrated environment.

CICD Pipelines

CICD

CICD = continuous integration (CI) and continuous delivery (CD) or, less often, continuous deployment (CD).

CI

Continuous integration (CI) is the practice of integrating source code changes frequently and ensuring that the integrated codebase is in a workable state.

CD - Delivery

Continuous delivery (CD) is a software engineering approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time.

CD - Delivery

It aims at building, testing, and releasing software with greater speed and frequency. The approach helps reduce the cost, time,[citation needed] and risk of delivering changes by allowing for more incremental updates to applications in production.

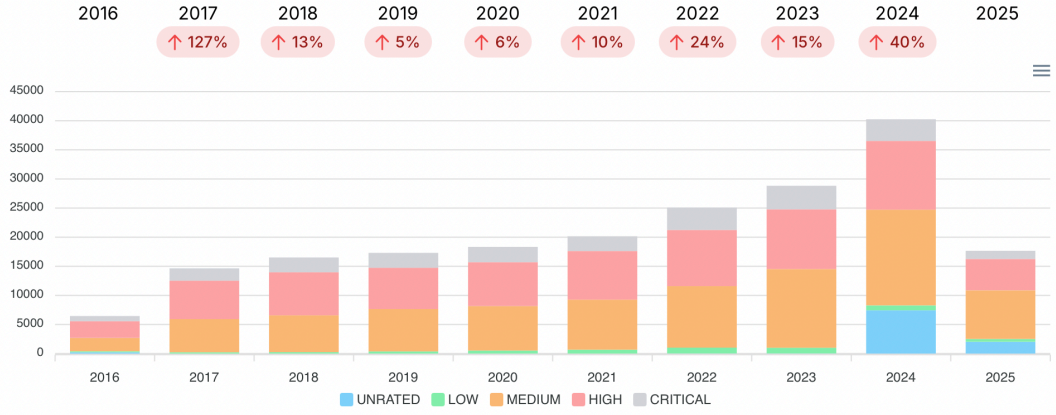
CD - Deployment

Continuous deployment (CD) is a software engineering approach in which software functionalities are delivered frequently and through automated deployments.

CD - Deployment

Continuous deployment contrasts with continuous delivery (also CD), a similar approach in which software functionalities are also frequently delivered and deemed to be potentially capable of being deployed, but are actually not deployed. As such, continuous deployment can be viewed as a more complete form of automation than continuous delivery

2016 - 2025 Vulnerability Severity By Year



Software Testing

Software testing can determine **the correctness** of software for specific scenarios but cannot determine correctness for **all** scenarios.

In other words, it cannot find **all** bugs.

Testing alone is NOT enough

Testing vs Proving

How can we proof our code (algorithm) is correct?

Formal methods

Formal methods

A collection of different math techniques to conduct technically and functional proofs

- mathematical guarantee of absence of bugs or vulnerabilities
- mathematical guarantee that the code functionally behaves as per its specs
- or manually executed

Formal methods

- Similar to Static Analysis
- Much deeper analysis
- or manually

Without **Formal methods** we cannot guarantee the correctness of our algorithm and code!