Secure Product Lifecycle

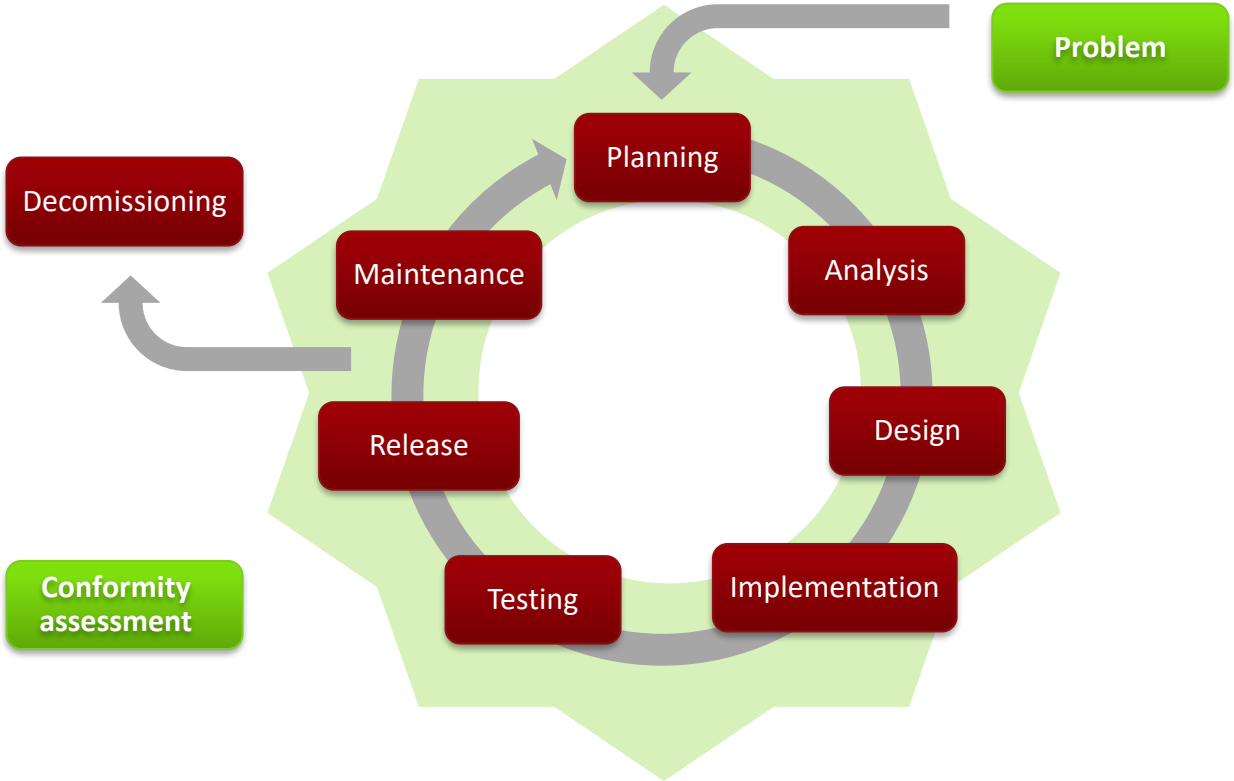
Security Testing

Yagoba GmbH, Austria, www.yagoba.com, info@yagoba.com

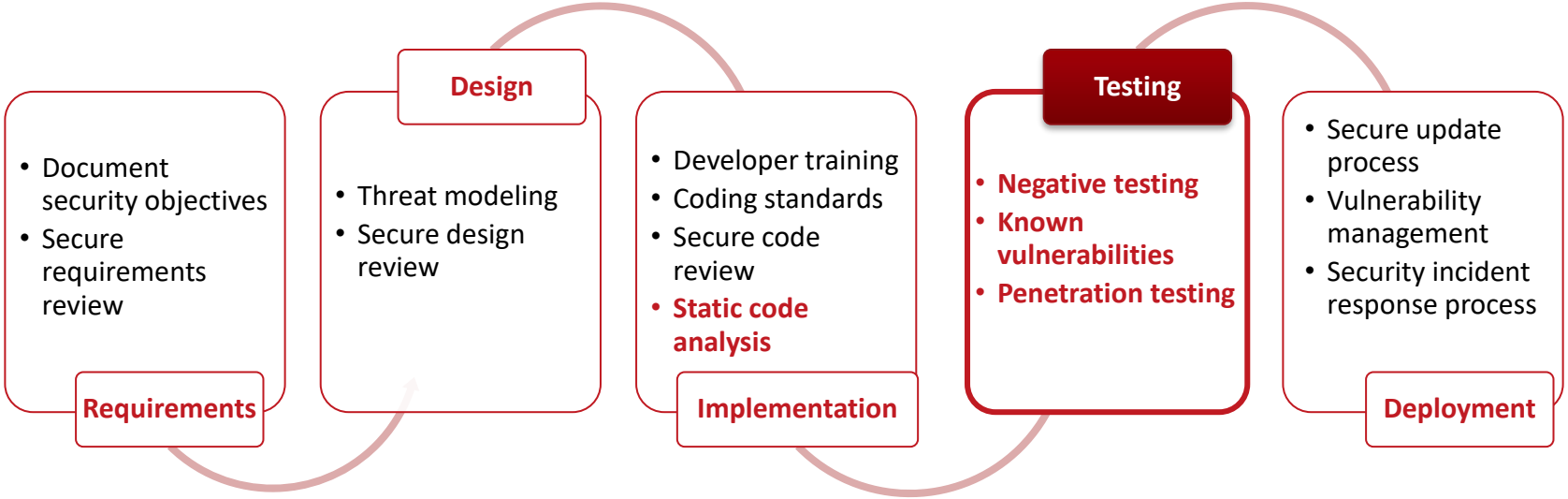
Today‘s Agenda

- Context within SDLC – Recap
- Security Testing
  - Functional Security Testing
  - Robustness Testing
- Attack Scenarios
- Software Testing Fundamentals

Secure **Product** Lifecycle



Secure **Development** Lifecycle



Functional Security vs. Robustness

- The secure feature „barrier“ is **functionally correctly** implemented…

Functional Security vs. Robustness

- The secure feature „barrier“ is functionally correctly implemented…

- …but the implementation is **not robust** against attacker not following instructions!

Functional Security vs. Robustness Testing

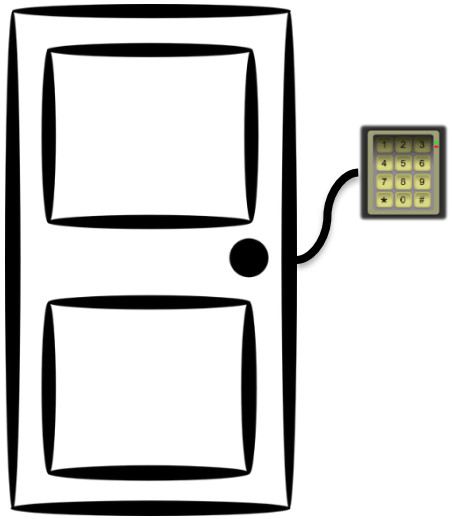
- Testing the security of products requires
  - Functional Security Testing and
  - Robustness Testing together.

- Functional Security Testing
  - Demonstration that your security is correctly implemented
  - Usually done via standard functional testing
- Robustness Testing
  - Demonstration that the way how your security is implemented cannot be bypassed
  - Usually done via penetration testing, fuzzing, etc.

Yagoba GmbH, Austria, www.yagoba.com, info@yagoba.com

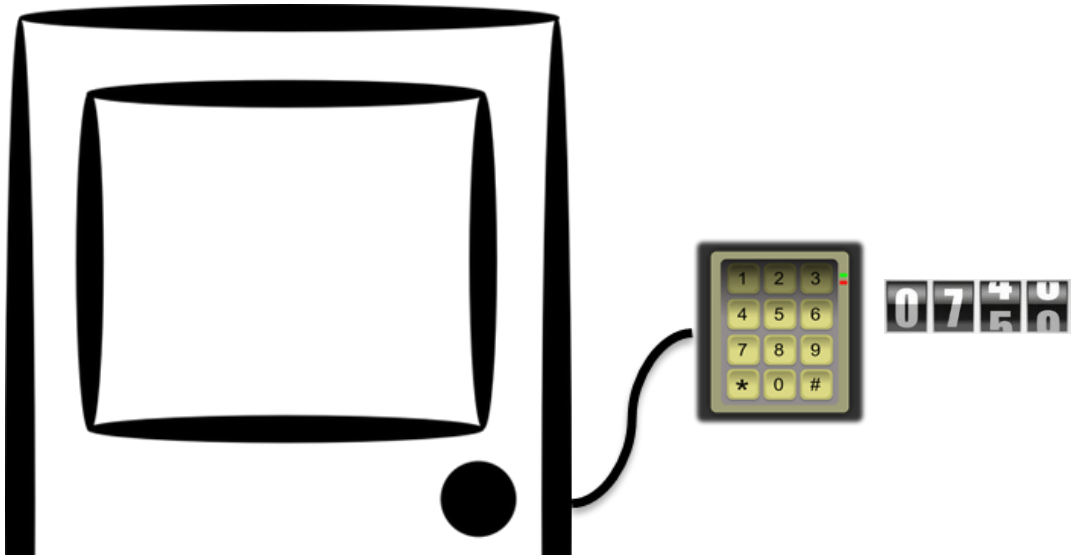
ATTACK PRINCIPLES

Attack Principles



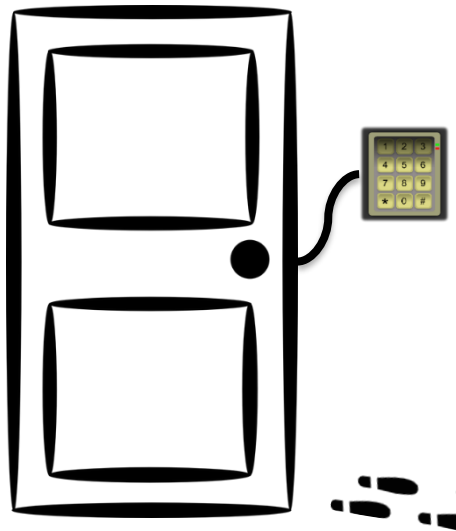
Attack Principles

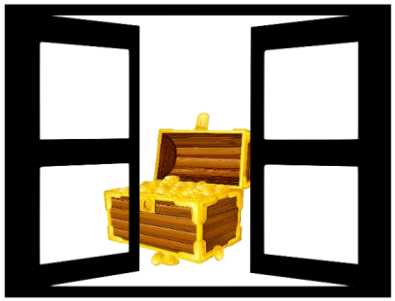
- Trying out all combinations via **BRUTE FORCE**



Attack Principles

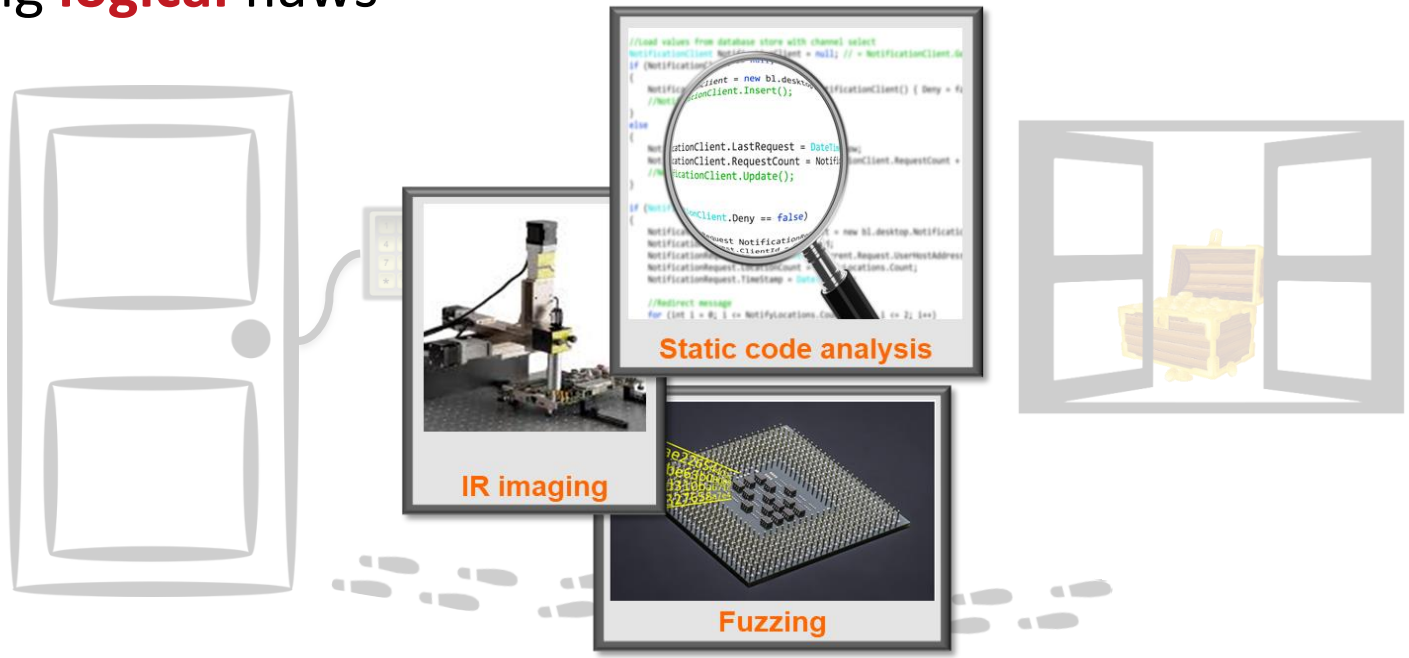
- Finding **logical** flaws





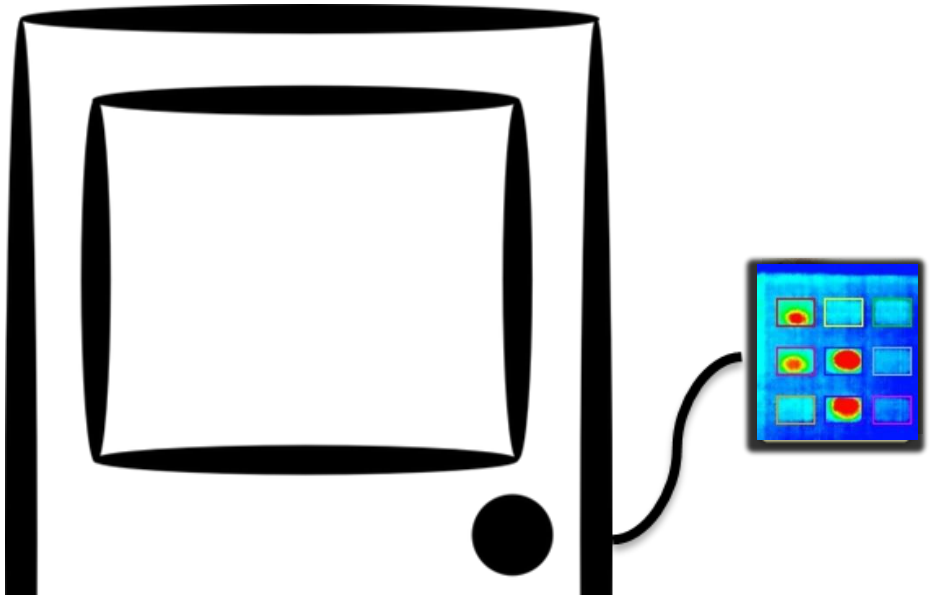
Attack Principles

- Finding **logical** flaws



Attack Priniciples

- Taking short cuts via **SIDE CHANNELS**



Attack Priniciples

- Taking short cuts via **SIDE CHANNELS**

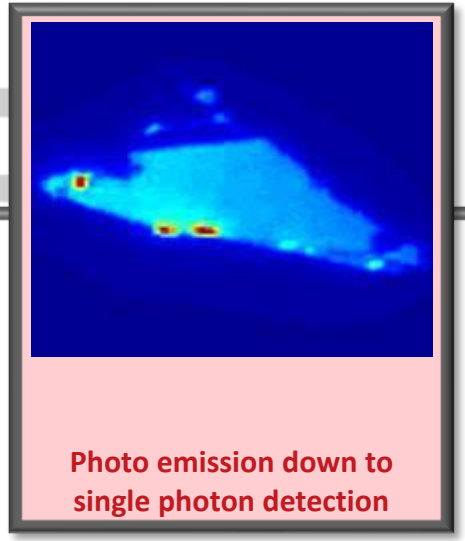
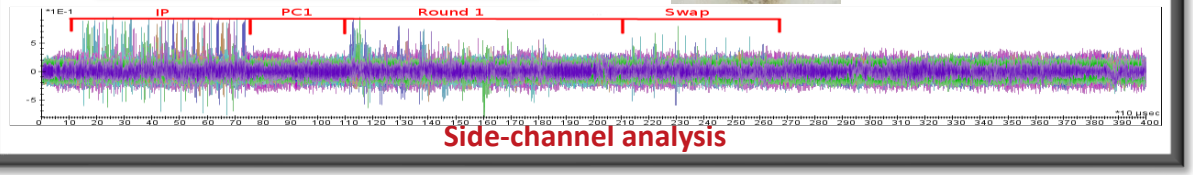


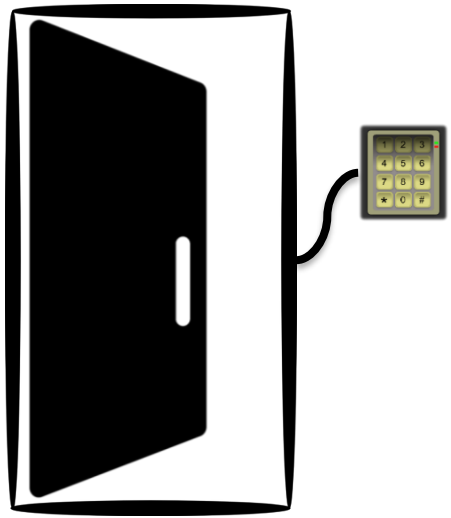
Photo emission down to single photon detection



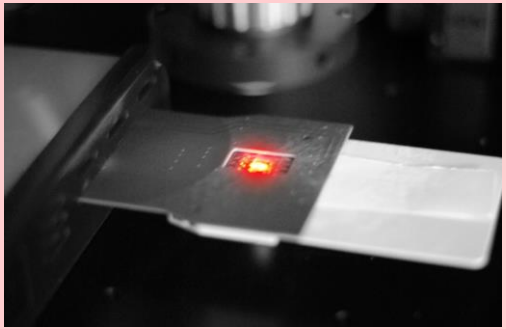
Side-channel analysis

Attack Principles

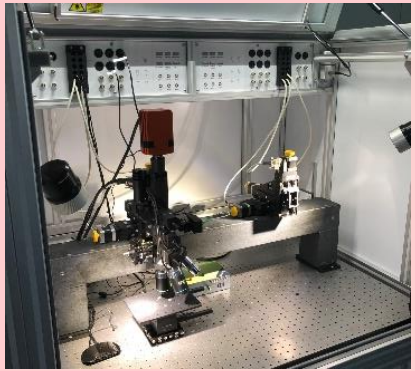
- Bypassing via Injection of **FAULTS**



Attack Priniciples



Laser Fault Injection



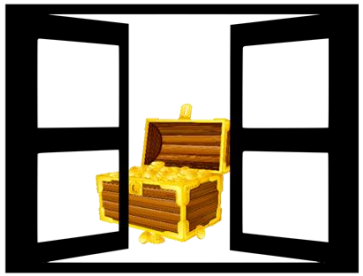
EM fault injection

- Bypassing via injection of **FAULTS**

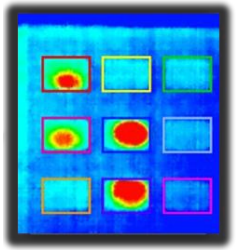
Attack Principles Summary

Brute Force

Logical



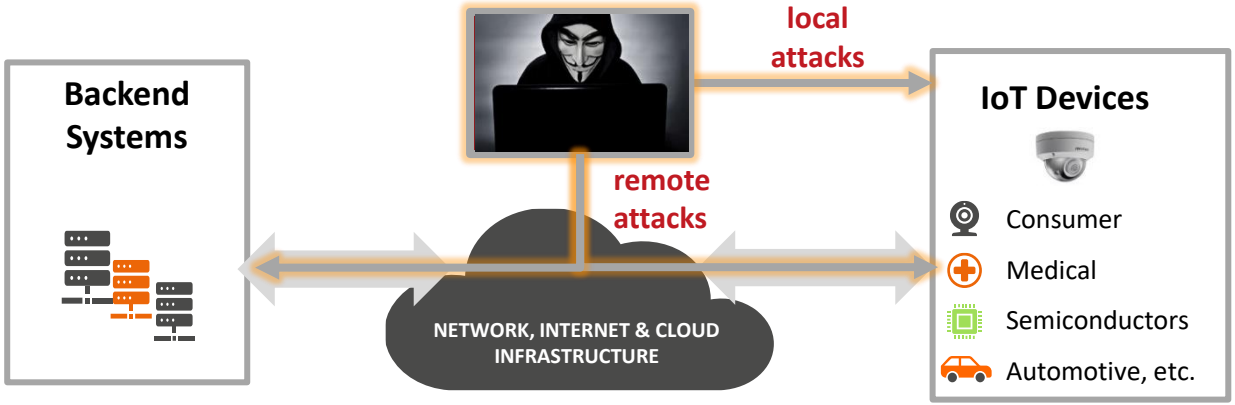
Side Channel



Fault

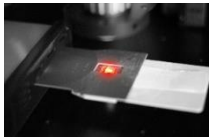
also called „physical attacks“

Attack Principles applied



local logical

e.g. test car maintenance interface



local physical

classical smartcard hacking

A successful attack often combines several attack principles. E.g. a local physical attack can be used to reverse engineer a device and by this the attacker learns how to attack multiple devices of the same type in the field by remote logical attacks.

remote logical

classical network hacking

remote physical

new category, where smartcard attacks are now applied remotely

SOFTWARE TESTING FUNDAMENTALS

Yagoba GmbH, Austria, www.yagoba.com, info@yagoba.com

Motivation

- Testing is integral to any software development lifecycle
- Testing identifies defects, errors, or bugs so that they can be fixed prior to the product’s release in the market
- Testing identifies if requirements are sufficiently fulfilled and security for your threat model and risk analysis is achieved
- Testing in the SPLC also facilitates evaluators and conformity assessment

Detour to good testing

- Increasing costs for defect management per lifecycle: testing early and in a structured way
  - „Shift Left Approach“
- Testing for security offers an even larger toolbox
- Testing shows presence, not absence of defects
- Don‘t fall for the Absence-of-defects failacy
  - Verification vs Validation

Software Testing Methods

- White box testing
  - Focus on how the system works, often also known as structural testing
- Black box testing
  - Focus on what the system does, (not how it‘s done), more outsider/end user perspective
- Grey box testing
  - Combining Black and Whitebox testing, having partial knowledge of internal workings

Whitebox Testing

- Testing internal behavior and structure
- „Full Knowledge“ or „Open Book“ Test

- Techniques/Tools:
  - Static Analysis
  - Debuggers
  - Code Coverage

- Benefits
  - Thoroughness
  - Optimization
  - Security
- Drawbacks
  - Time-consuming and complex
  - Not suitable for large code bases
  - Requires skilled resources

Blackbox Testing

- Testing the product without knowledge about the internal workings
- „Zero Knowledge“ Test, simulate „Outsider“ Attack
- Techniques:
  - Equivalence Partioning
  - Boundary Value Analysis
  - Decision Table Testing

- Benefits
  - No need for internal knowledge
  - Unbiased testing
- Drawbacks
  - Limited coverage
  - Inefficiency in identifying certain types of issues

Greybox Testing

- Everything between Black- and Whitebox testing
- Only some partial knowledge (e.g. Credentials)
- Most commonly used type

- Techniques/Tools:
  - Combine all from Black- and Whitebox
  - Regression Testing

- Benefits
  - More coverage than with whitebox-testing
  - Beneficial if functional and structural aspects should be tested
- Drawbacks
  - Limited coverage compared to whitebox testing
  - Inefficiency in identifying certain types of issues

Types of Tests

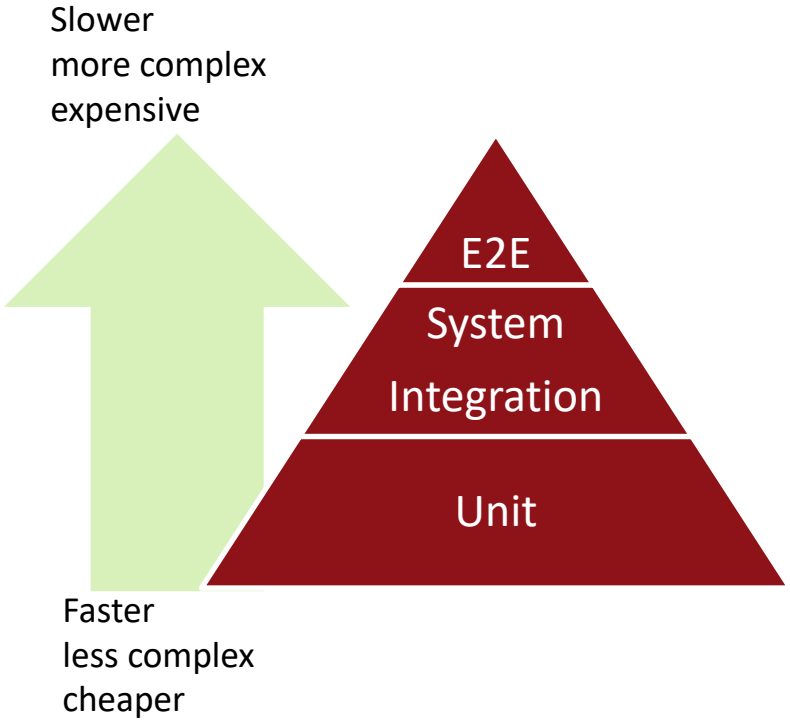
International Software Testing Qualification Board (ISTQB) distinguishes between:

- Dynamic tests are the tests carried out by executing the code
- Static tests for which the code is not executed (code analysis, linting, compliance with coding rules, code reviews, etc.)

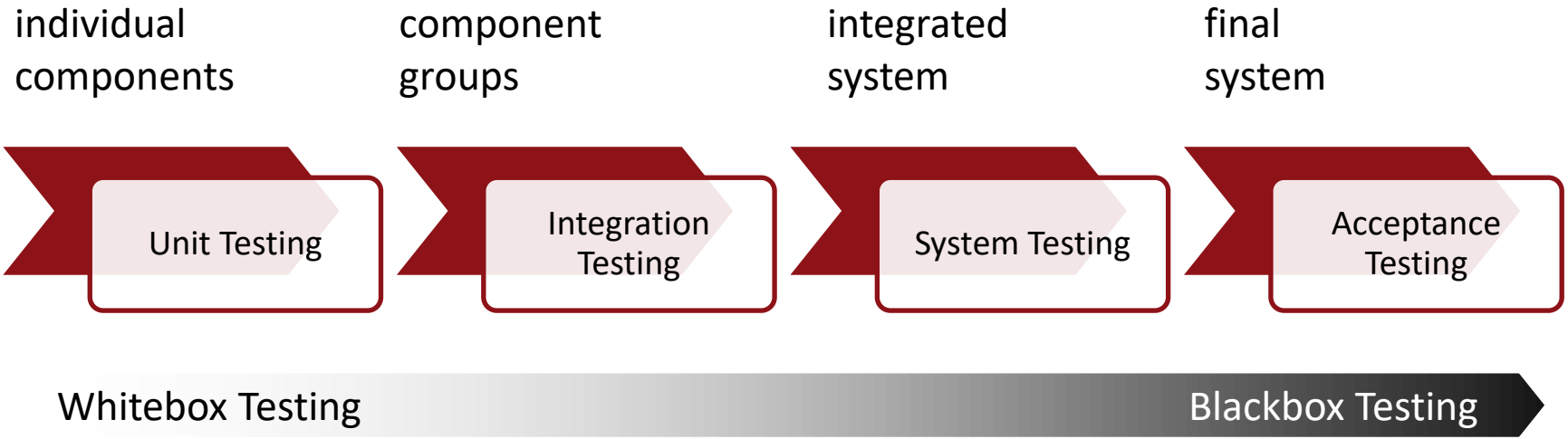
Test Levels/Test Pyramid

- Unit Tests
- Integration Tests
- Acceptance/End to End

- Further seperate categories like
  - Stress, Performance, …



Test Pyramid



Glossary

- Target of Evaluation (TOE)
  - A set of software, firmware, and/or hardware components that are the subject of a security evaluation process. The TOE defines the boundaries of the product or system to be tested, including its security functions, interfaces, and implementation.
- Device under Test (DUT)
  - Also Test Target, Object under Test. Defines the particular part that is the focus of testing.
- Test Plan
  - A document describing the scope, approach, resources and schedule of intended test activities. It is a record of the test planning process.
- Test (Case) Specification
  - A document specifying a set of test cases (objective, inputs, test actions, expected results, and execution preconditions).

Test Plan

- A Test Plan describes the objectives, ressources and processes for a test project
- Content of a testplan can include:
  - Scope, Test objective, constraint
  - Assumptions
  - Stakeholders (roles, responsibilities, training needs)
  - Risks
  - Test approach (Test levels, Test Types, Deliverables, Metrics,)
  - Budget or schedule

- More info on via ISTQB or ISO/IEC/IEEE 29119-3

Test Specification

- Let‘s remember STRM
- Also the Test (Case) Specification should provide the traceability for security requirements
- The specification is written **BEFORE** the testimplementation
- Derive from Security Requirements your structure
  - Test Classes, Test Groups, structure allows for detecting missing topics

Example Test Case Specification

TestID	Req ID	Req Focus	Topic	Objective	Pre-Condition	Test Procedure	Pass Criteria	Added	Last Modified	Test Case Status
FUN-10-03-01	R22	Perso	App Perso-nalization	Send command with surplus valid data	PRE3	1. For all valid options of the data: 1.1 Send the command “CSR” with this additional valid data part in random order	An error must be returned with code 0x0020	1.0.0	1.3.1	Defined
FUN-10-10-00	R22, R40	Perso	App Perso-nalization	Check the state for a new Vehicle ID	PRE2	1. Provision an Longterm Key Pair 2. Read the STATUS flag via Diagnostic Read command 3. Send the command “CSR” with valid data and option 0x9A 4. Verify STATUS flag	All steps must pass and return SW 9000 and the STATUS has changed in from step 2 to step 4 to indicate 0x02/42	1.0.0	1.3.0	Defined

Positive vs Negative Testing

- Positive Tests reflect the „happy path“, the good case, a use case
- Negative Tests reflect the unexpected, invalid input, the misuse or absuse case and therefore attack scenarios
- Positive Testing is not enough for Security Testing
  - Even if you have fulfilled all defined user requirements, is your product secure?
  - With the bottom up approach of having security considered in the requirements, you stand a better chance

Example

- Assume a method/command: `doSomething(parameter1);`
- Investigation/having specification at hand, we see that `parameter1` can have three valid input values of type int: 0x00, 0x01 and 0x10.
- What tests do we write for that?
  - Test any value?
  - Test the three valid ones? 0x00, 0x01, 0x10
  - Test the three valid ones and another (which one?)
  - Just test all the possible input values?

Example: Equivalence class

- Valid: 0x00, 0x01, 0x10

Equivalence classes	Values
0x00	0x00
0x01	0x01
0x02-0x0F	0x02, 0x0F, Random(0x03, 0x0E)
0x10	0x10
0x11 – 0xFF	0x11, 0xFF, Random(0x11, 0xFF)

Example: Equivalence Class

- Not all test types necessarily show the same output/behaviour

Equivalence classes	Values
0x00	Positive Testing
0x01	Positive Testing
0x02-0x0F	Negative Testing
0x10	Positive Testing
0x11 – 0xFF	Negative Testing

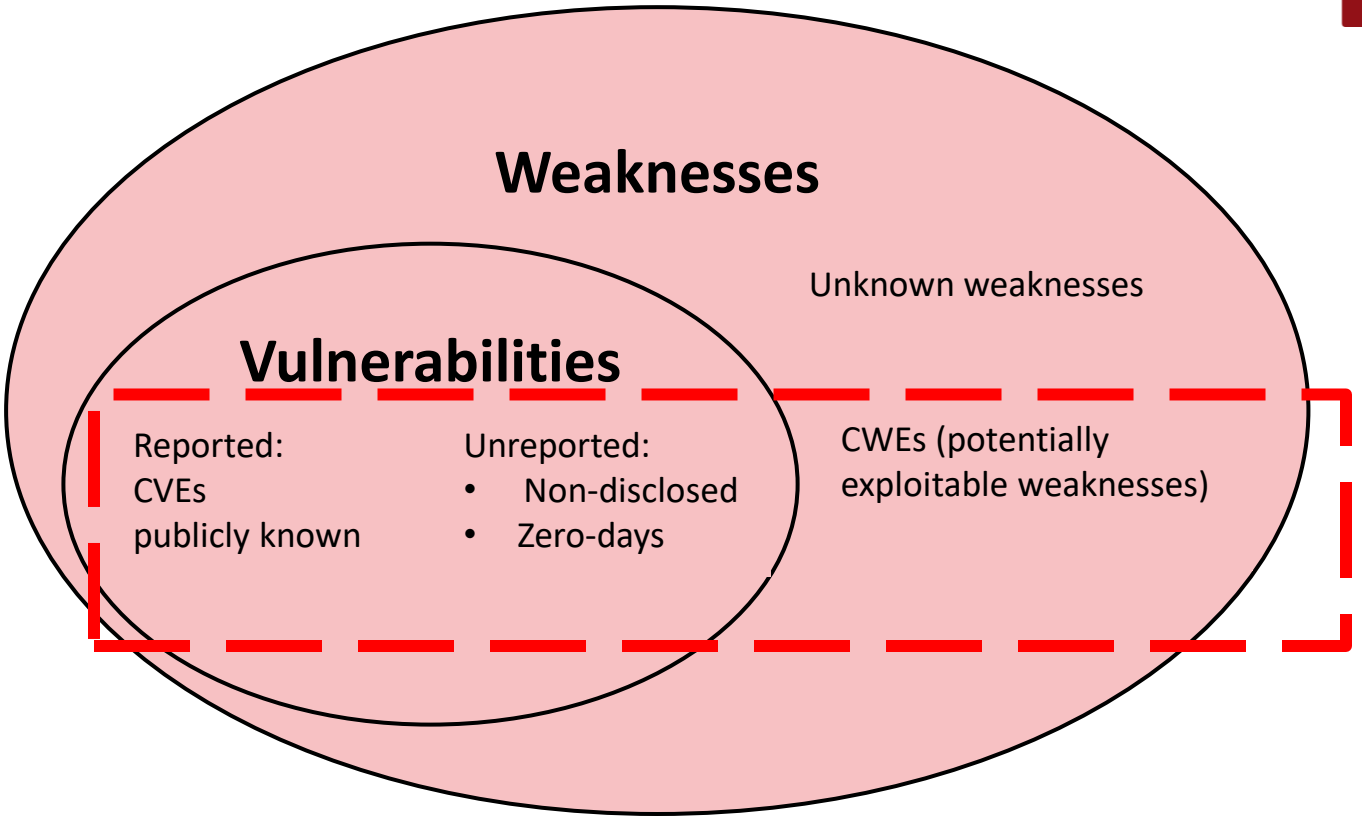
How to reflect change in testing

- Confirmation testing
  - Introduced change/bug fix works as intended
  - New test cases will be defined
- Regression Tests
  - Introduced change does not have no negative side effects on unchanged parts
  - Already existing tests might need adaption
- Check the whole SPLC if updates are necessary to e.g. requirements or your threat model/risk analysis

Detour end – back to Security Testing

- Automate your testing
- Automate your negative testing, e.g. with Fuzz Testing
- For Robustness, also consider these parts of security testing:
  - Vulnerability Scanning
  - Penetration Testing
  - Security Audits
  - Social Engineering Tests
  - Security Review

Recap


Weaknesses
Unknown weaknesses
Vulnerabilities
Reported:
CVEs
publicly known
Unreported:
• Non-disclosed
• Zero-days
CWEs (potentially exploitable weaknesses)

Known Vulnerabilities

- Testing 3rd party software might not easily testable (Blackbox)
- Vulnerability scanning is an automated, high-level test that looks for and reports potential known vulnerabilities.
- https://owasp.org/www-community/Vulnerability_Scanning_Tools

SUMMARY

Summary

- Testing for Functional Security and for Robustness
- Attack Scenarios and how they can be grouped depending on the exploited attack surface
- Fundamentals of Software Testing, not only for Security
  - Types of Testing
  - Test Level
  - Example of Test Design via Equivalence Class Approach

LV Timetable

	Topic	Date	Time	Lecturer
	Introduction & Overview	09.10.2024	12:15-13:45	Christoph Herbst
1	Risk Analysis & Threat Modeling	16.10.2024	12:15-13:45	Christoph Herbst
2	Secure Design & Requirements Management	23.10.2024	12:15-13:45	Karin Maier
3	Security Testing	30.10.2024	12:15-13:45	Karin Maier
4	**Fuzz Testing**	06.11.2024	12:15-13:45	Srđan Ljepojević
5	**Penetration Testing: Web and Mobile App**	13.11.2024	12:15-13:45	Tomislav Nad
6	**IoT and Device Security Testing**	20.11.2024	12:15-13:45	Christoph Herbst
7	**Security Testing: Implementation attacks**	27.11.2024	12:15-13:45	Christoph Herbst
8	**Conformity assessment**	04.12.2024	12:15-13:45	Christoph Herbst
9	**Security from release to decommissioning**	11.12.2024	12:15-13:45	Christoph Herbst