



Automotive Cybersecurity Validation Strategy

2020-12-10 | Nico Vinzenz | ZF Friedrichshafen AG



Why do we need an Automotive Validation Strategy?

Obligatory “Jeep in the ditch”-Picture

- **Jeep Cherokee Hack**
 - 2015 by Miller and Valasek



But the List goes on..



TESLA



Tesla (2015): Remote vehicle unlock and start



HYUNDAI



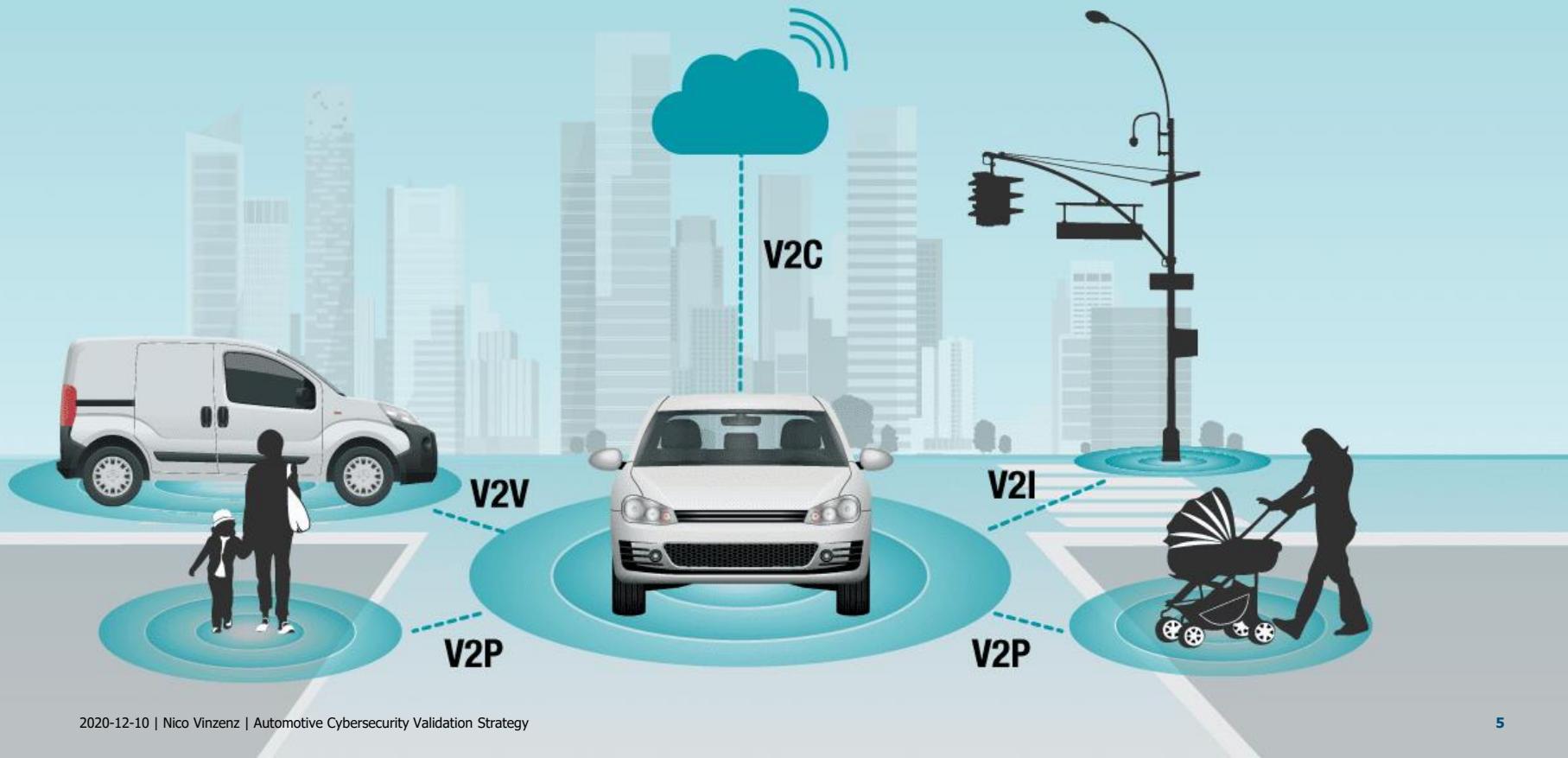
Mercedes-Benz

Nissan (2016): Remote instrument panel control

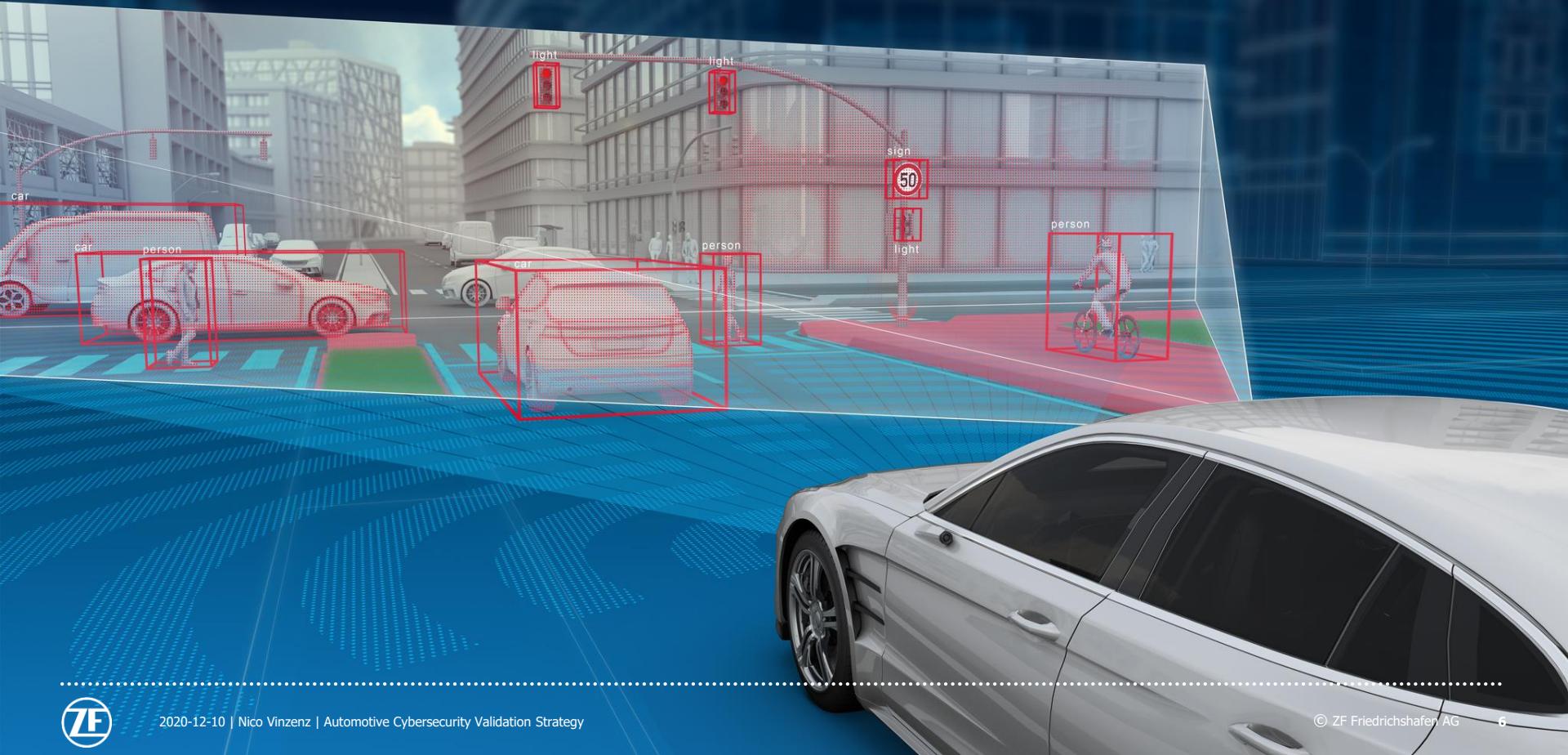
Hyundai (2017): Remote vehicle unlock and start

Mercedes-Benz (2019): Remote vehicle unlock and start

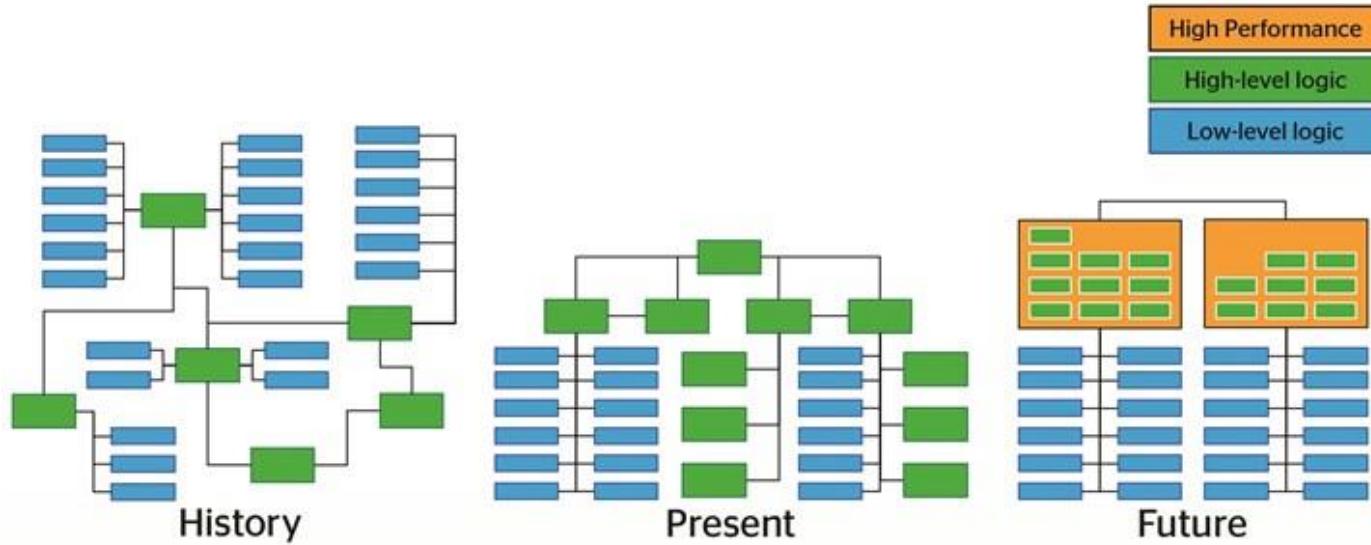
Emerging Challenge – V2X Connectivity



Emerging Challenge – AD Functionality



Emerging Challenge – E/E Architecture



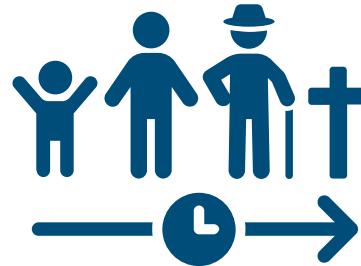
Further Cybersecurity Challenges



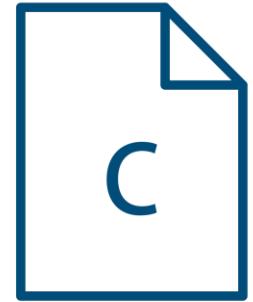
Competitive
Profit Margins



Hardware
Restrictions



Lifetime of
15+ Years



Insecure
Programming
Languages

Agenda

- 01 Automotive Development Process**
- 02 Security Validation Strategies**
- 03 Practical Application**
- 04 Reducing Risks in the Future**

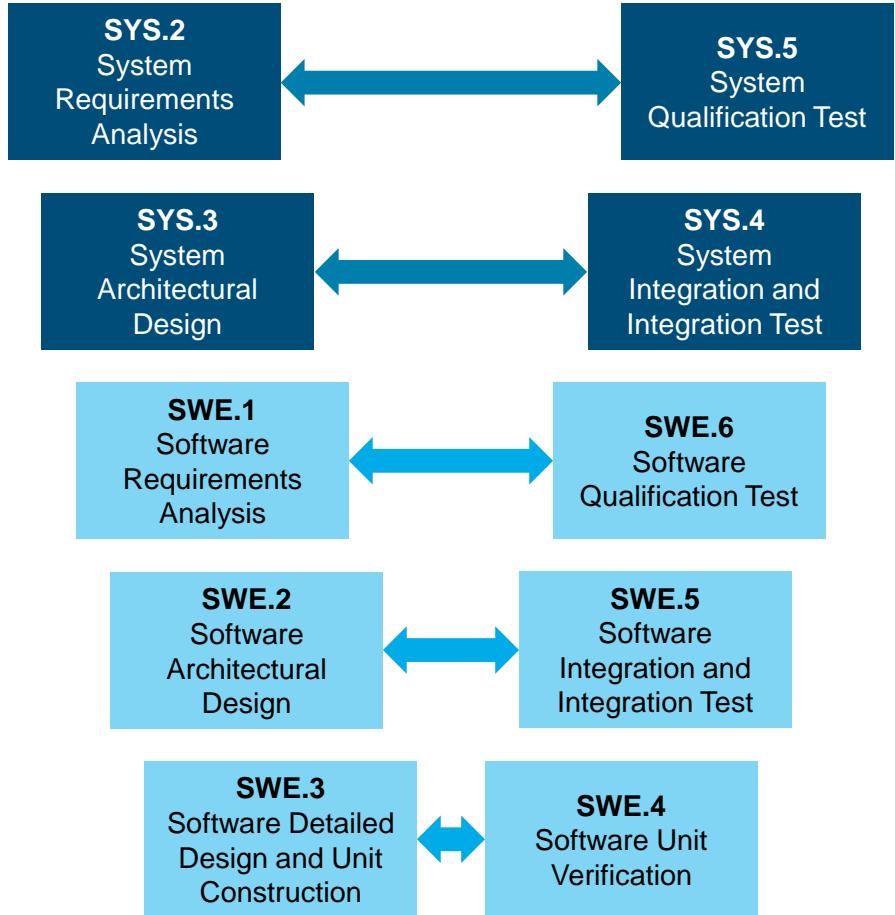


01

Automotive Development Process

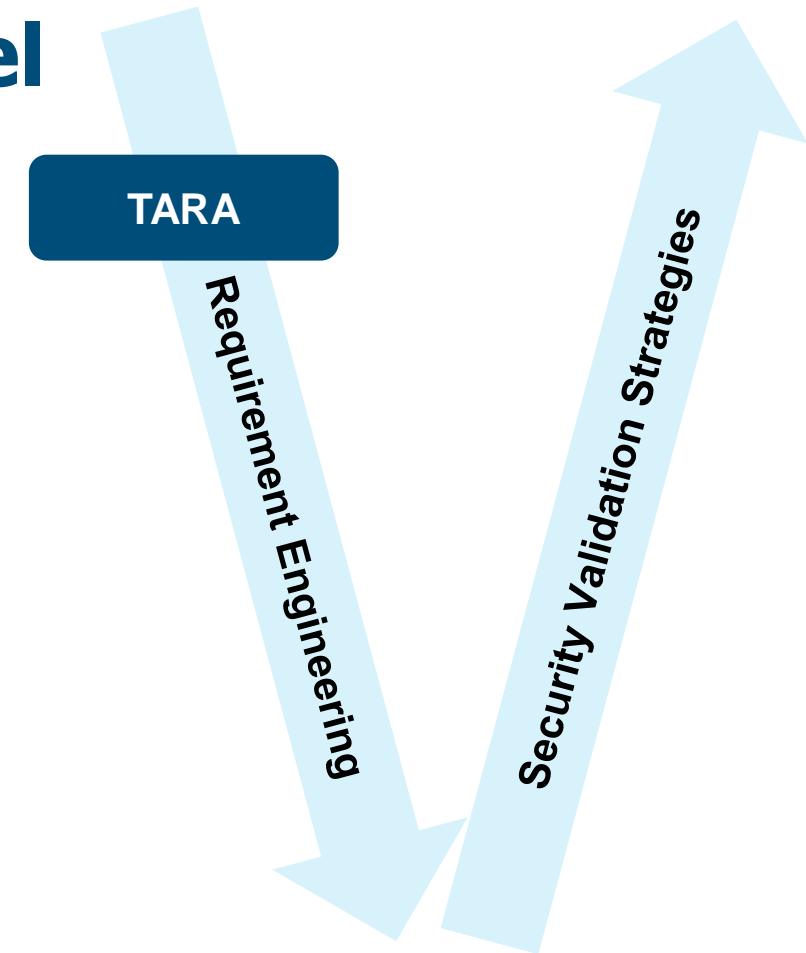


Standard V-Model



Cybersecurity V-Model

- **Threat Analysis and Risk Assessment (TARA)**
 - Input: System Assets
 - Output: Security Goals
- **Security Validation Strategies**
 - Validate Security Goals



02

Security Validation Strategies



Overview

TARA

Requirement Engineering

Security Validation Strategies

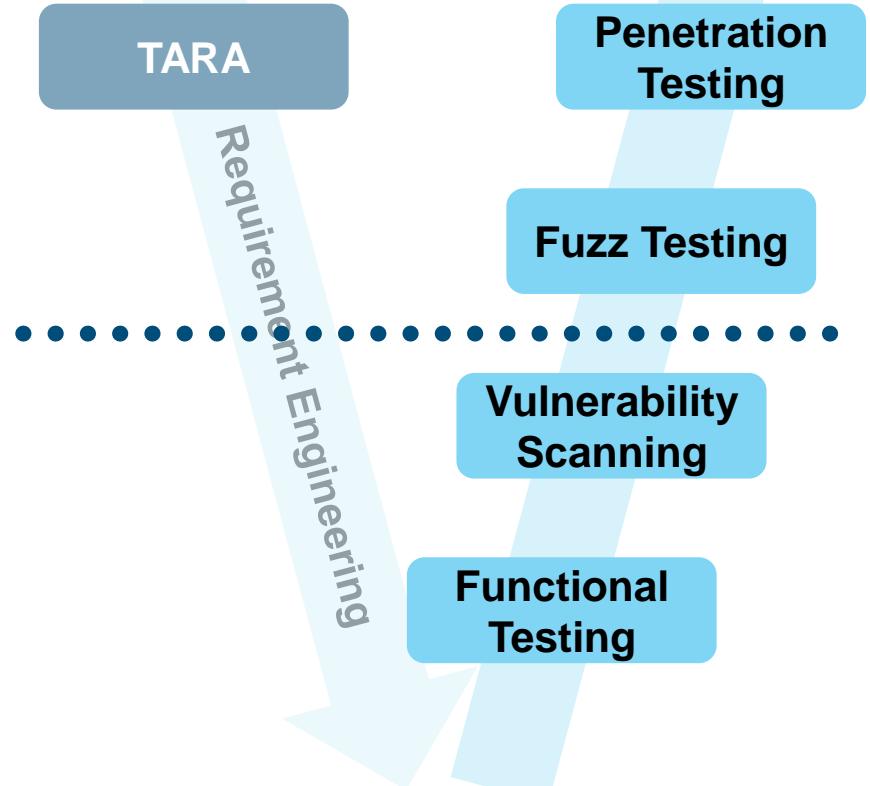


Overview

**Testing against
the “unknown”**

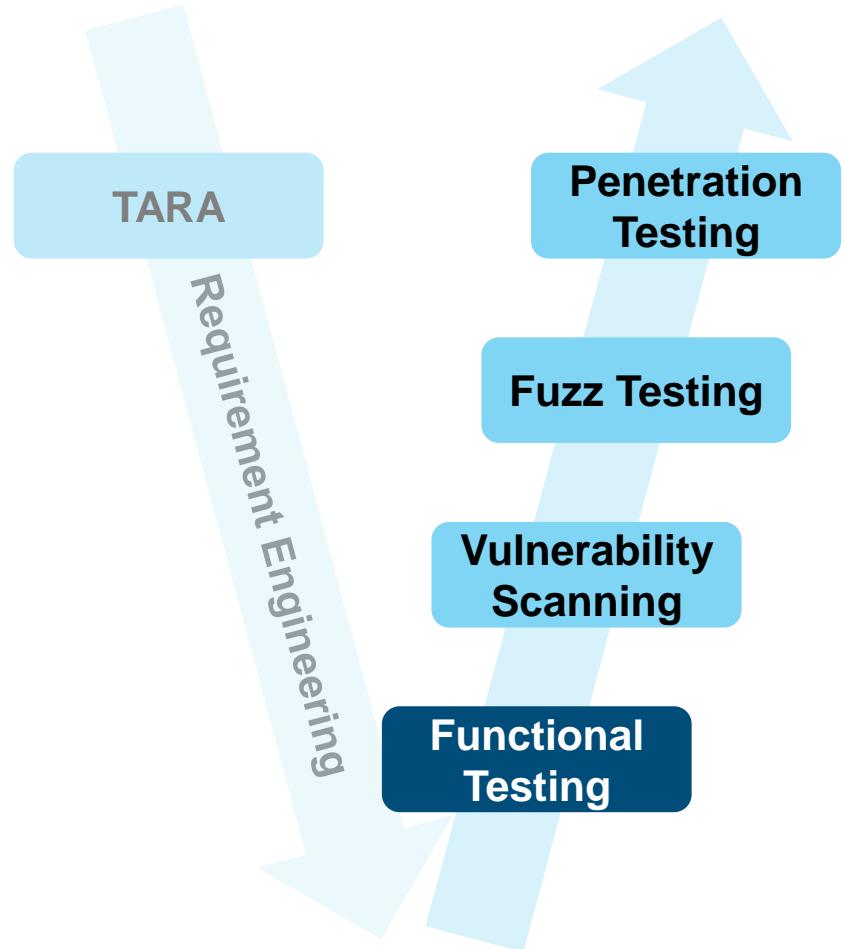


**Testing against
the “known”**



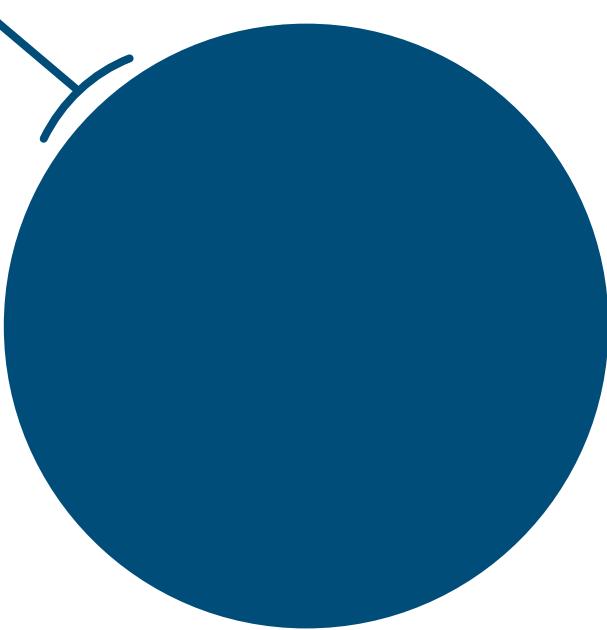
Functional Testing

- **Requirement-based Approach**
 - Translate functional requirements into test cases
 - Easy to find intended but not implemented behavior
 - Hard to find implemented but not intended behavior

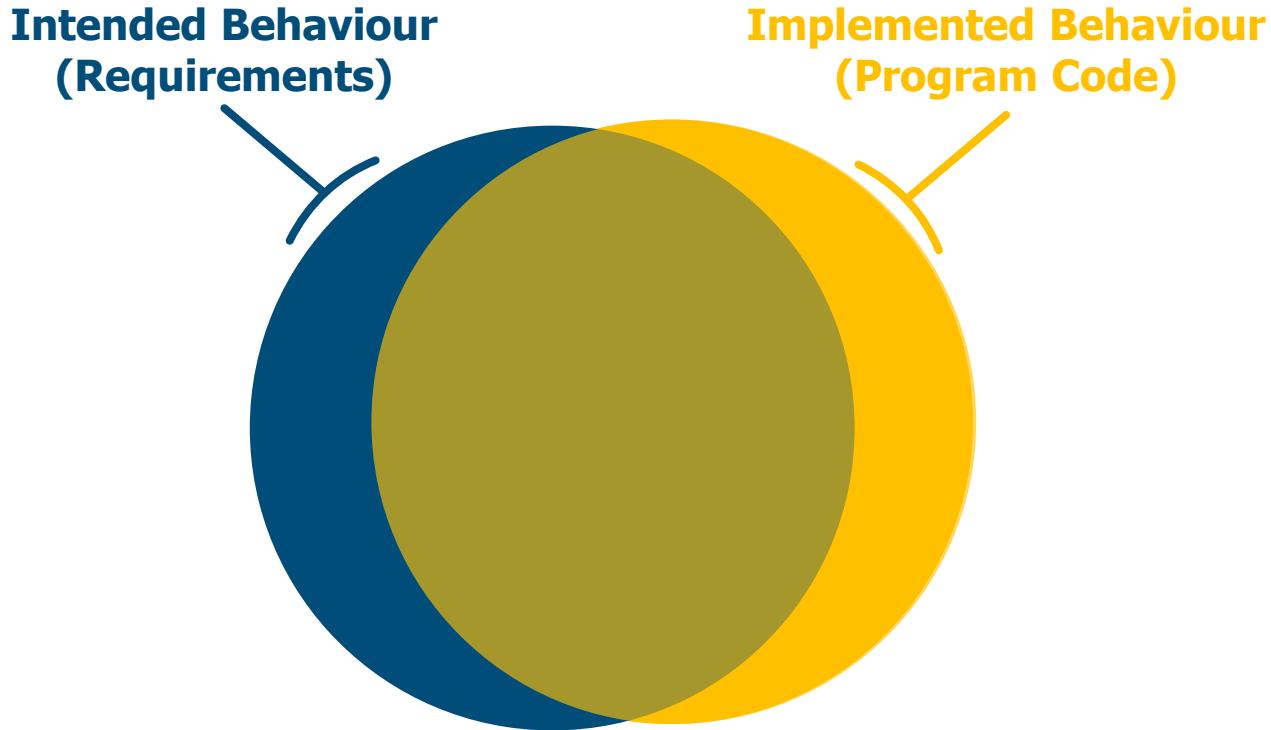


Functional Testing

Intended Behaviour
(Requirements)



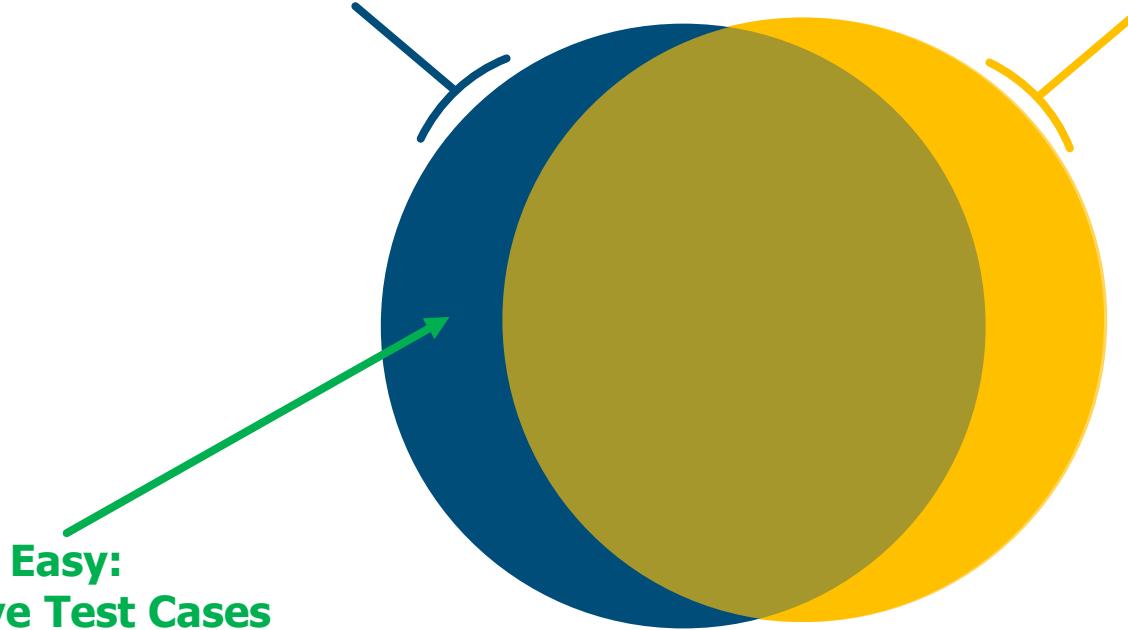
Functional Testing



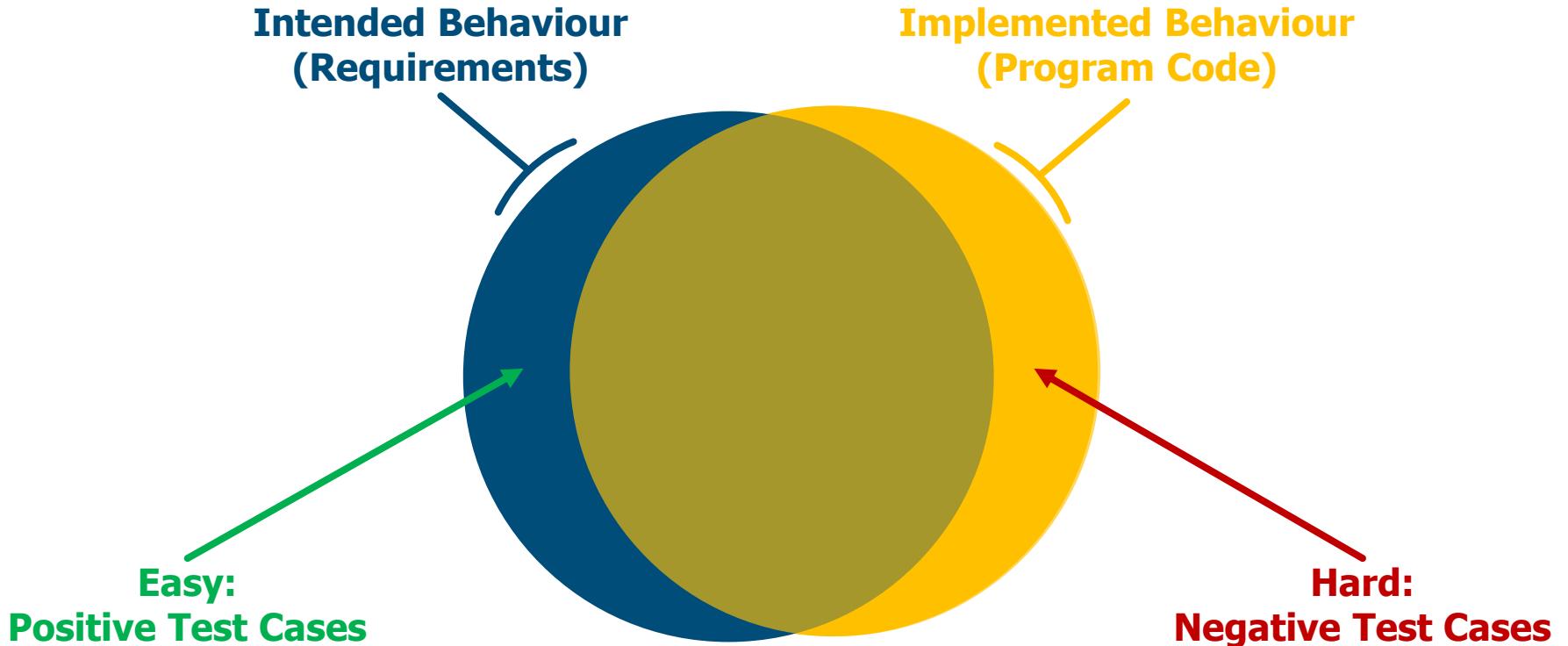
Functional Testing

Intended Behaviour
(Requirements)

Implemented Behaviour
(Program Code)



Functional Testing



Vulnerability Scanning

- **Knowledge-DB Approach**

- Static/dynamic code analysis finds weaknesses
- CVE-matching on Bill of Materials (BoM) finds vulnerabilities
- Port scanner finds configuration mistakes
- Oblivious to zero-day and product-unique exploits

The diagram illustrates the relationship between Requirement Engineering and several testing approaches. A large blue arrow labeled "Requirement Engineering" points downwards from the top center towards the bottom right. Six light blue rounded rectangles are arranged around this arrow, each containing a different testing method. Starting from the top left and moving clockwise, the methods are: TARA, Penetration Testing, Fuzz Testing, Vulnerability Scanning, Functional Testing, and another instance of TARA at the top left.

TARA

Penetration
Testing

Fuzz Testing

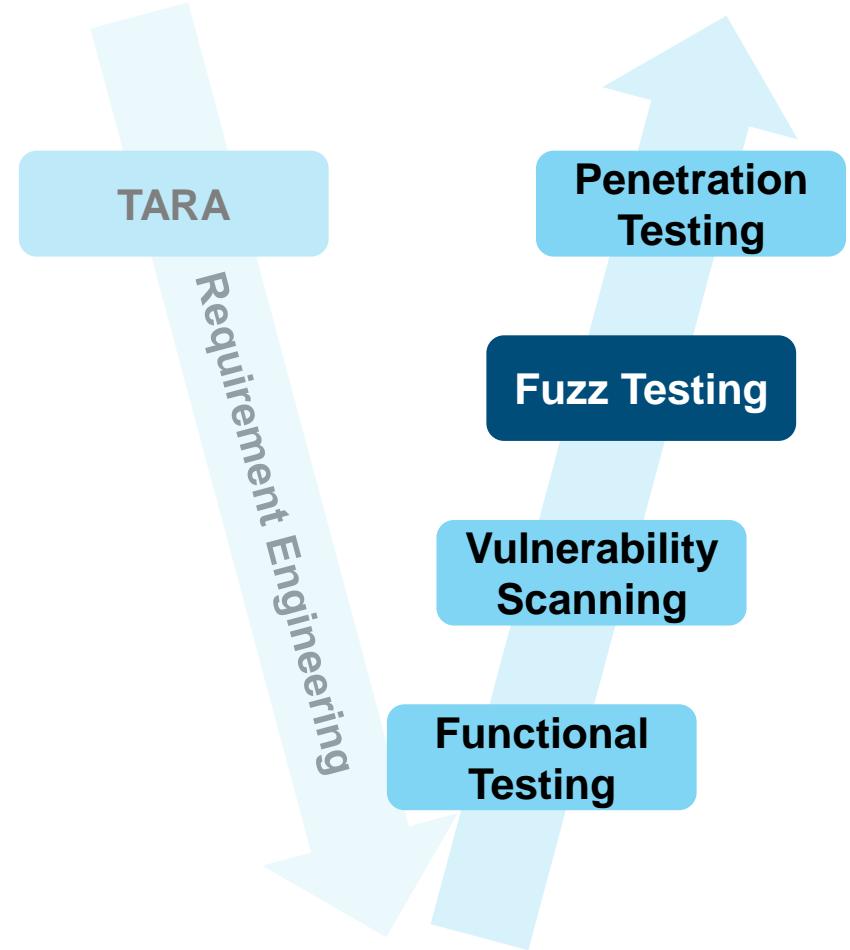
Vulnerability
Scanning

Functional
Testing

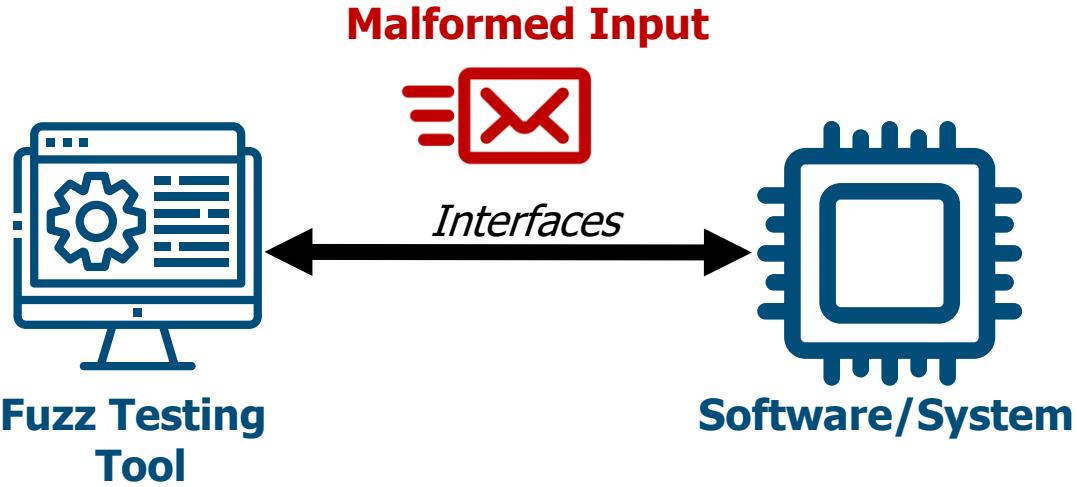
Fuzz Testing

- **Testing into the Void**

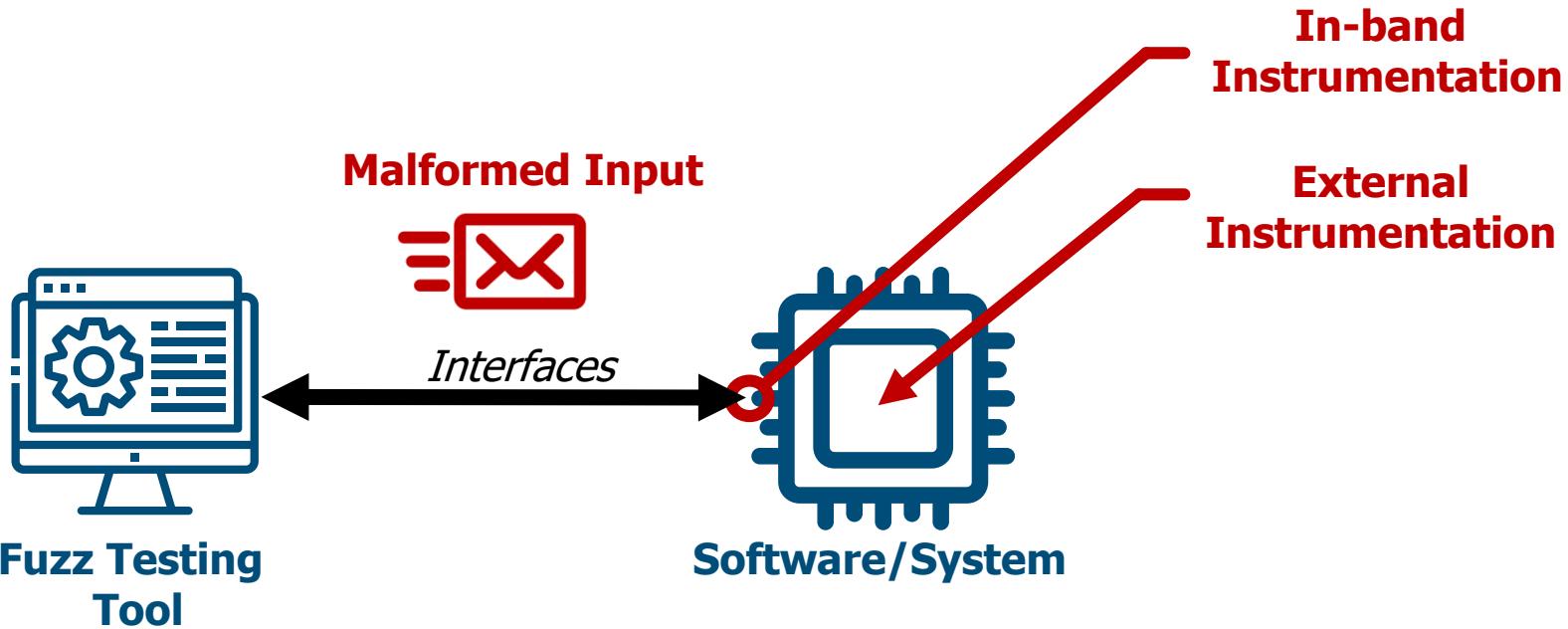
- Interface supplied with semi-valid data
- System monitored for suspicious behavior
- Can find unknown vulnerabilities
- Challenging to configure correctly and generate “evidence”



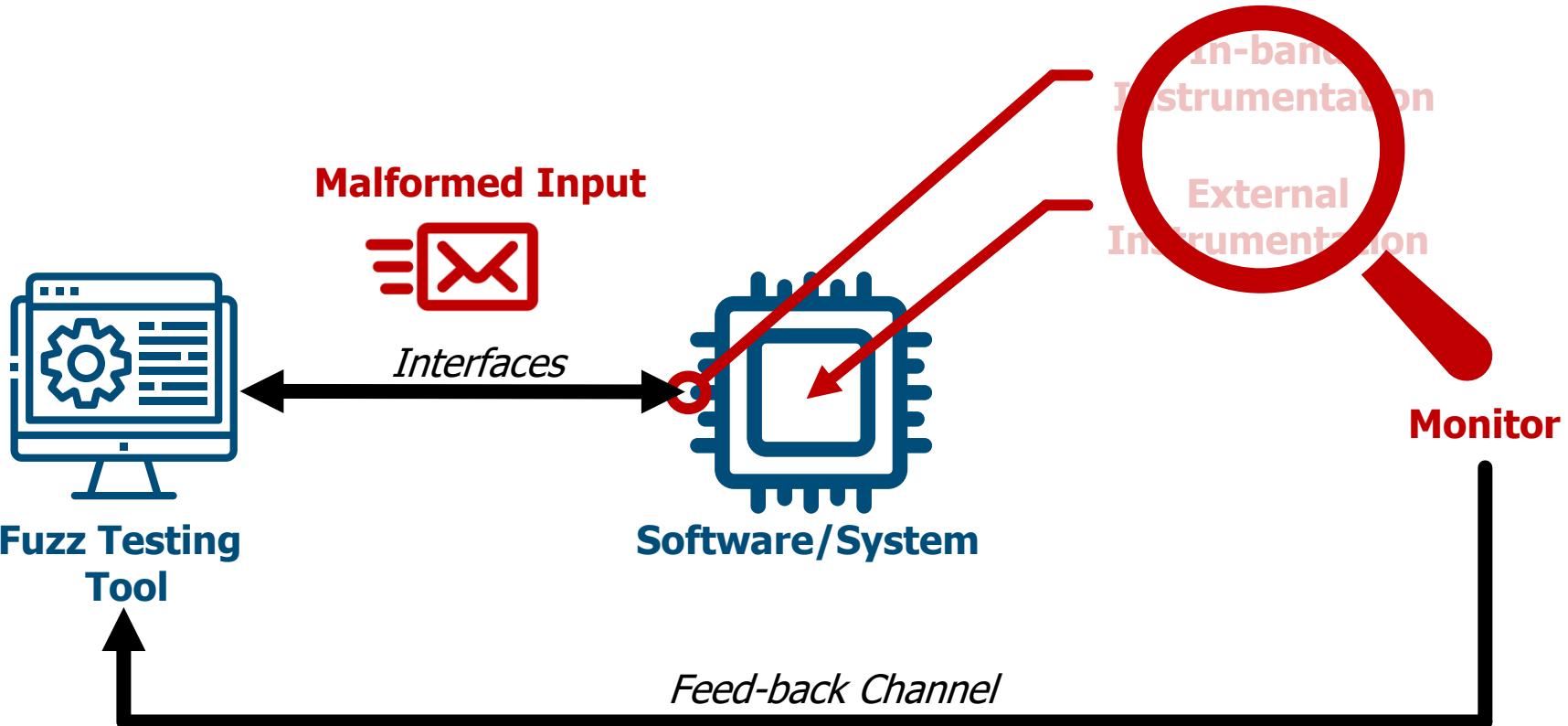
Fuzz Testing



Fuzz Testing



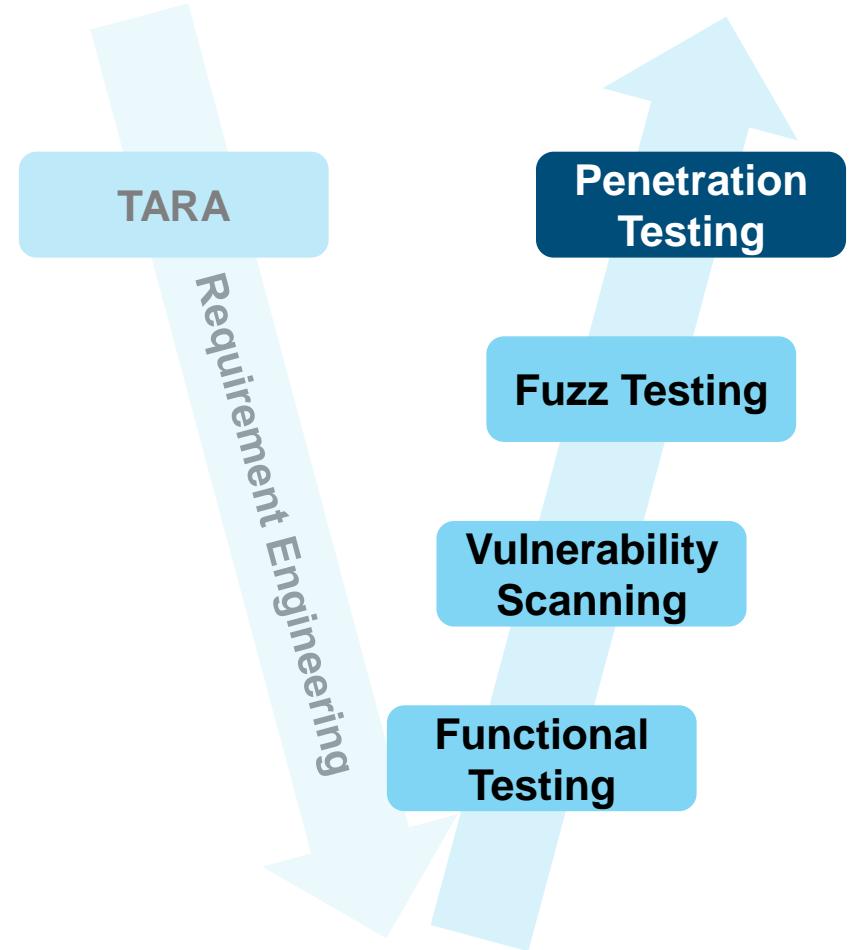
Fuzz Testing



Penetration Testing

- **Authorized Cyberattack**

- Security expert tries all available techniques within scope
- Can find unknown vulnerabilities
- High demand on effort and expertise makes it expensive



Penetration Testing



Large Budget

- Multiple experts
- High-tech lab for SCA
- Months of work

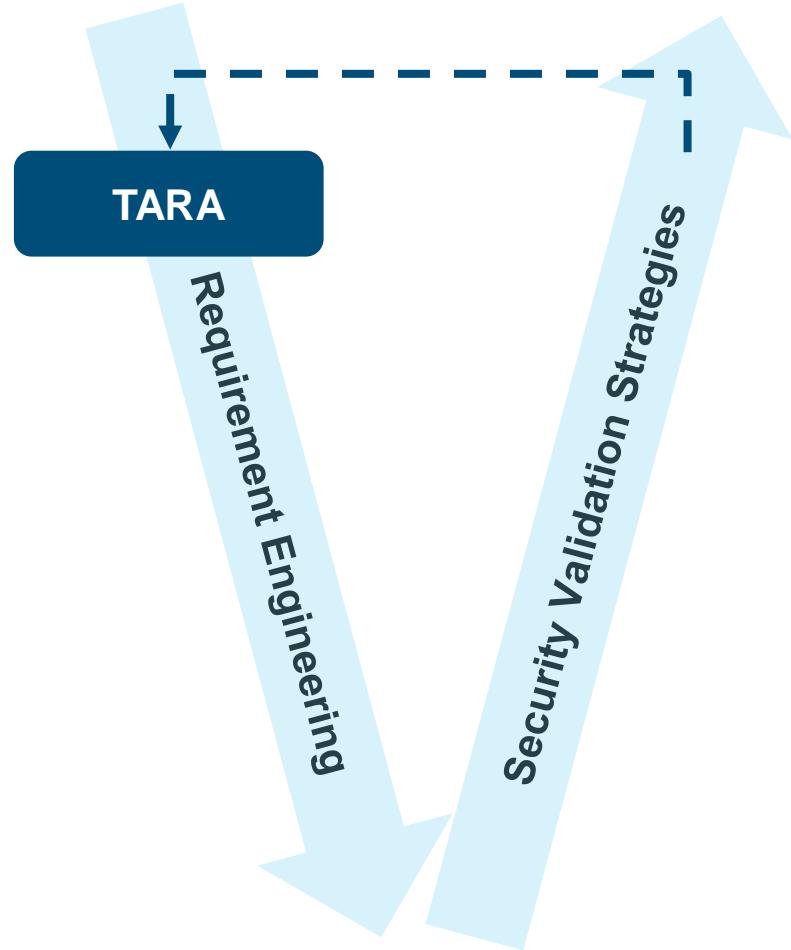
Small Budget

- “Script kiddies”
- Generic vulnerability scanner/fuzzer
- Days of work

Iterative Process

- **Iterative Process**

- 1. Findings flow back into the TARA
- 2. Adaption of security goals and requirements
- 3. Adaption of security validation strategies



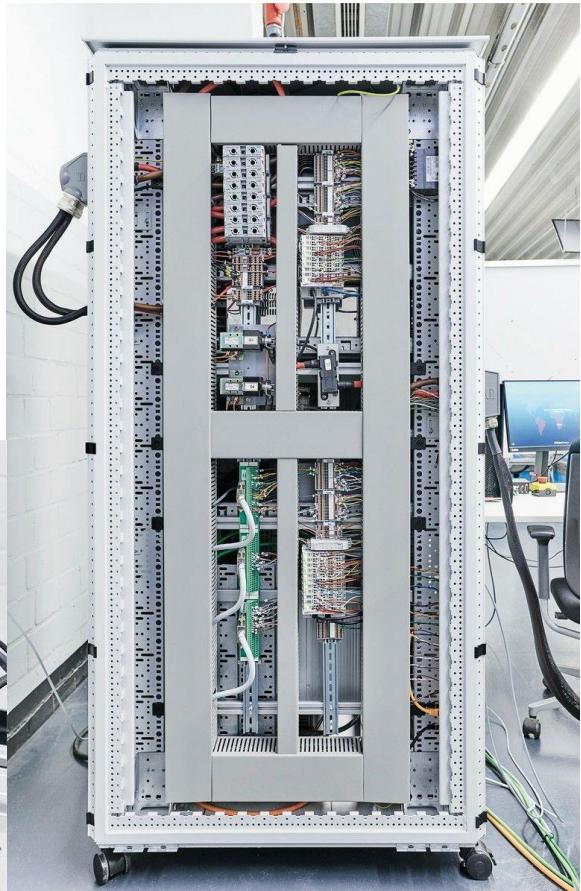
03

Practical Application



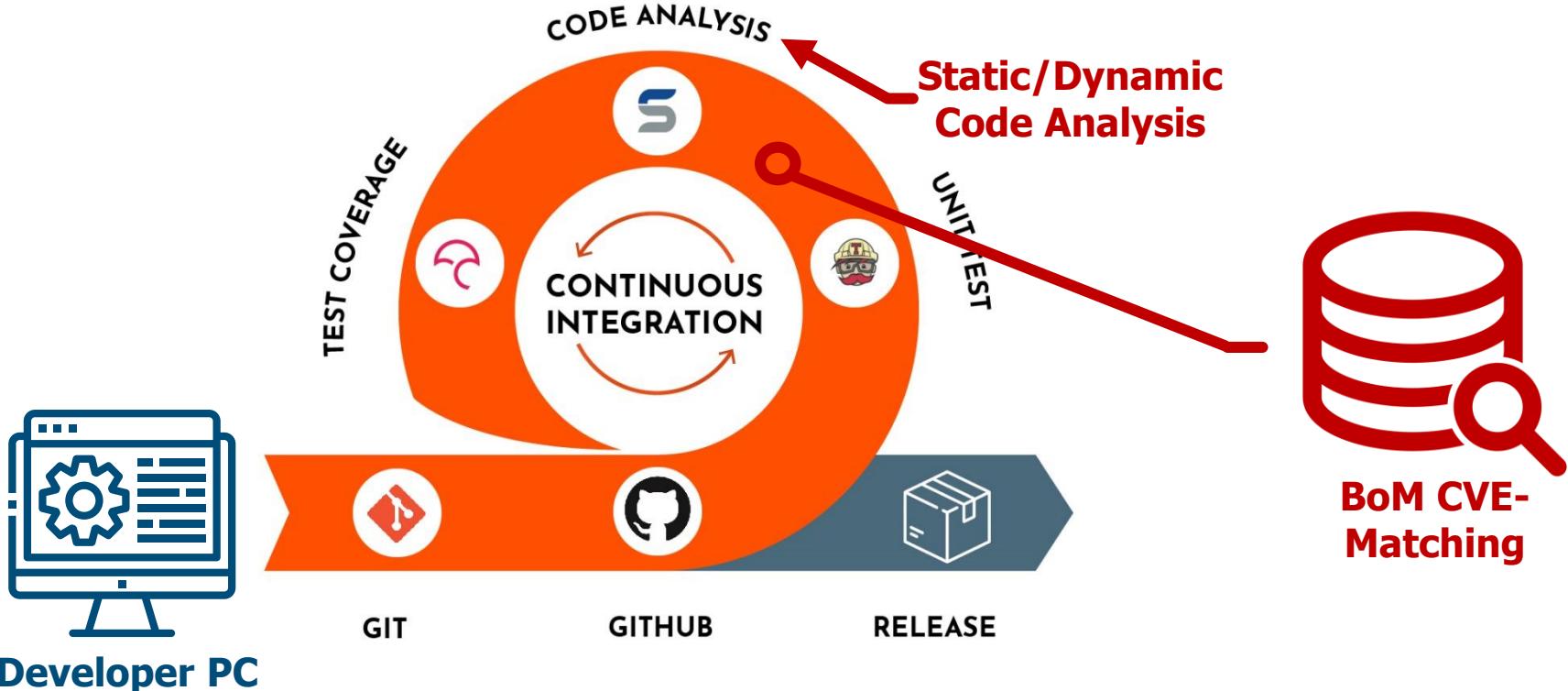
Functional Testing

- Execute test cases on ...
 - SiL (Software in the Loop)
 - HiL (Hardware in the Loop)



Hardware in the Loop (HiL)

Vulnerability Scanning



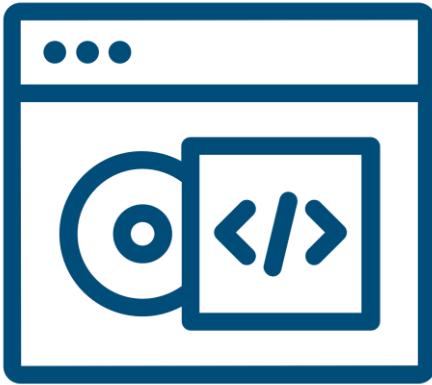
Vulnerability Scanning



Incident Response Process

- Interface to supplier and “Responsible Disclosure” researcher
- PSIRT (Product Security Incident Response Team) ensures appropriate reaction

Fuzz Testing



Software Unit Fuzzing

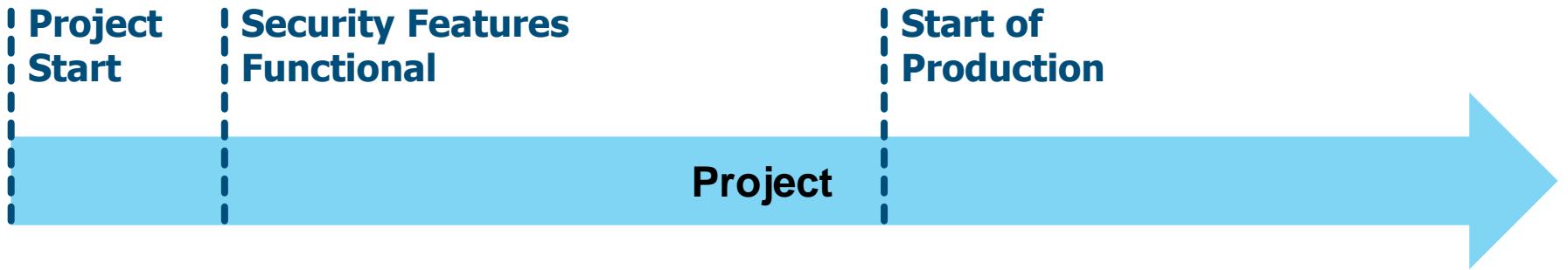
- X.509 certificates
- Custom written parser
- Data input
- Config files



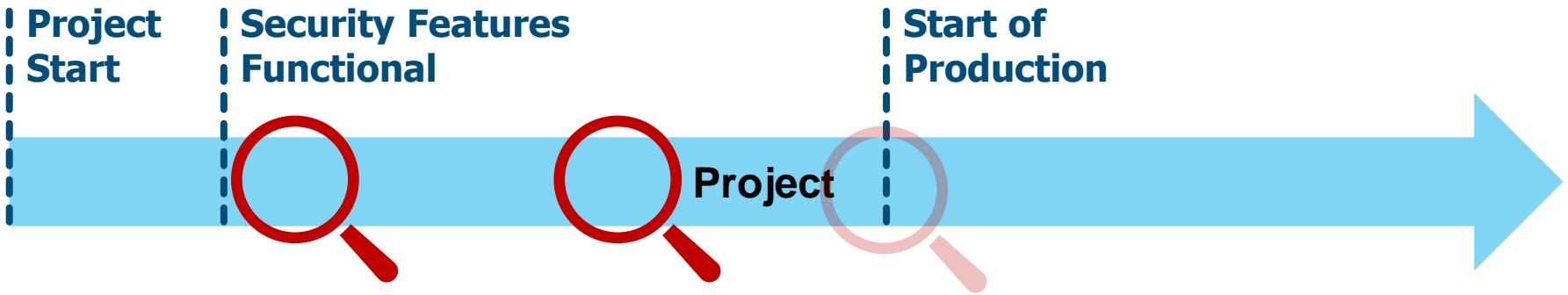
System Interface Fuzzing

- CAN (FD)
 - UDS
- Ethernet Stack
- IP, TCP, TLS/DTLS

Penetration Testing



Penetration Testing



04

Reducing Risks in the Future



Reducing Risks in the Future

- Upcoming regulation
 - UNECE WP.29 and ISO/SAE 21434
- CEP (Cybersecurity Engineering Process)
- Holistic Cybersecurity Concept
- LTS (Long Time Support)



Thank you!

Questions?

nico.vinzenz@zf.com

ZF Friedrichshafen AG behält sich sämtliche Rechte an den gezeigten technischen Informationen einschließlich der Rechte zur Hinterlegung von Schutzrechtsanmeldungen und an daraus entstehenden Schutzrechten im In- und Ausland vor.

ZF Friedrichshafen AG reserves all rights regarding the shown technical information including the right to file industrial property right applications and the industrial property rights resulting from these in Germany and abroad.

