Fraunhofer FOKUS
Institute for Open Communication Systems

Security Testing Approaches

System Testing and Validation Workshop (STV19)

QRS Conference

Martin A. Schneider | July 24th 2019 | Sofia, Bulgaria

Fraunhofer FOKUS

Agenda

1 Introduction

2 Static Security Testing

3 Dynamic Security Testing

4 Outlook

Introduction

1

What is Security?

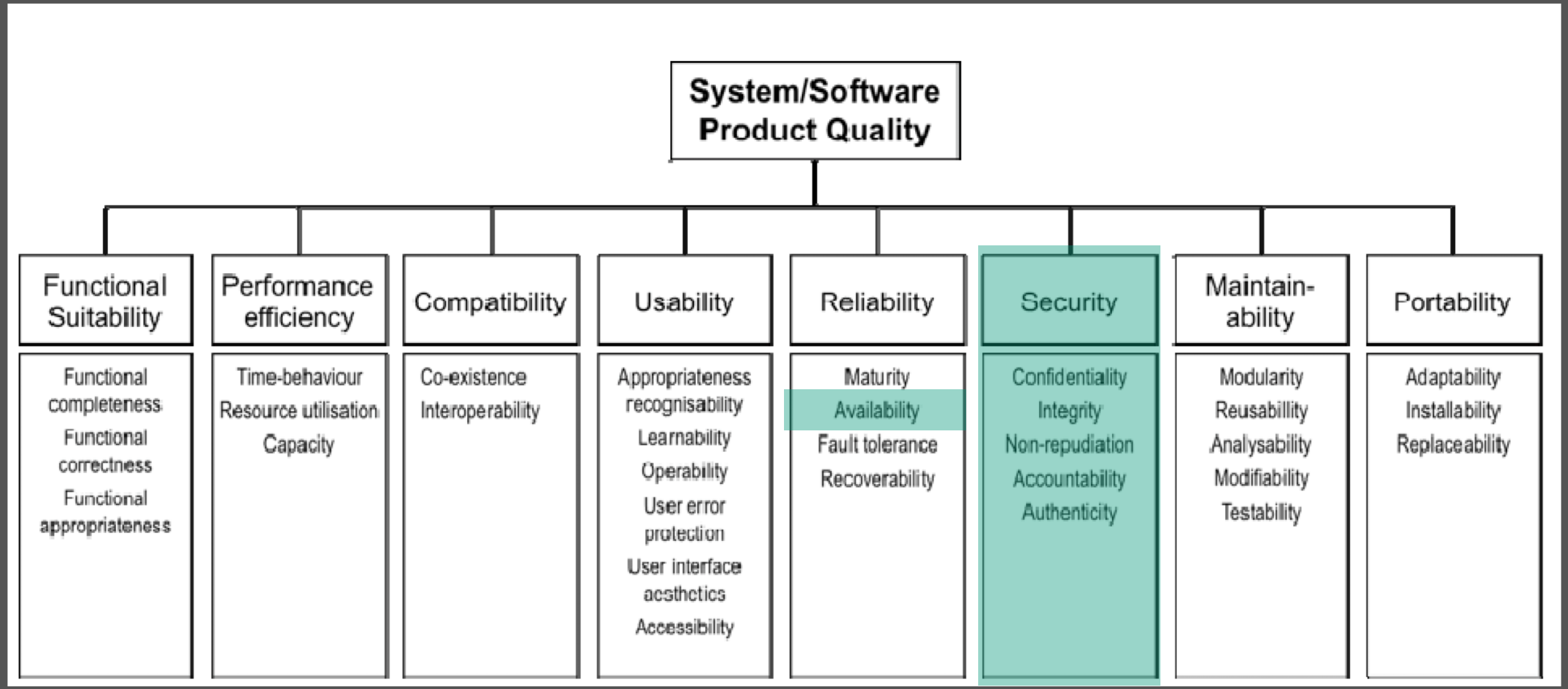
CIA

C = Confidentiality

I = Integrity

A = Availability

What is Security?



Source: ISO/IEEE 25010 (2011) System and software quality models

Fraunhofer FOKUS

Introduction: Security Testing

“Type of testing conducted to evaluate the degree to which a test item, and associated data and information, are protected so that unauthorized persons or systems cannot use, read, or modify them, and authorized persons or systems are not denied access to them.”

IEEE 29119 (2013) Software and systems engineering – Software testing – Part 1: Concepts and definitions

“Testing to determine the security of the software product.”

ISTQB Glossary v3.2.1 (2019)

“[…] include:

- Risk Assessment and Risk-based Security Testing
- Functional Testing of Security Features
- Performance Testing
- Robustness Testing
- Penetration Testing”

ETSI TR 101 583 V1.1.1 (2015-03): Methods for Testing and Specification (MTS): Security Testing - Basic Terminology

Introduction: Vulnerability

“Weakness of an asset or control that can be exploited by one or more threats“

ISO/IEC 27005 (2011) Information technology – Security techniques – Information security risk management

“A weakness in the system that could allow for a successful security attack.”

ISTQB Glossary v3.2.1 (2019)

“any weakness in software that can be used to cause a failure in the operation of the software”

ETSI TR 101 583 V1.1.1 (2015-03): Methods for Testing and Specification (MTS): Security Testing - Basic Terminology

Introduction

Black-box security testing

testing without reference to the internal structure of test item

White-box security testing

testing based on an analysis of the internal structure of the test item

Gray-box security testing

having some internal information of the test item

Static security testing

test item **is not executed** whilst testing it for the non-functional quality property security

Dynamic security testing

test item **is executed**, e.g. a component, software or system

Static Security Testing

2

Static Security Testing

Reviews

Rule checking

Abstract interpretation

Symbolic execution

Model checking

Dynamic Security Testing

3

Dynamic Security Testing

Functional Testing of Security Features

Performance Testing

Robustness Testing ⇨ Fuzz Testing

Penetration Testing

Fuzzing

fuzzing is about injecting invalid or unexpected inputs

– to obtain unexpected behavior

– to identify errors and potential vulnerabilities

interface robustness testing

fuzzing is able to find (0day-) vulnerabilities, e.g.

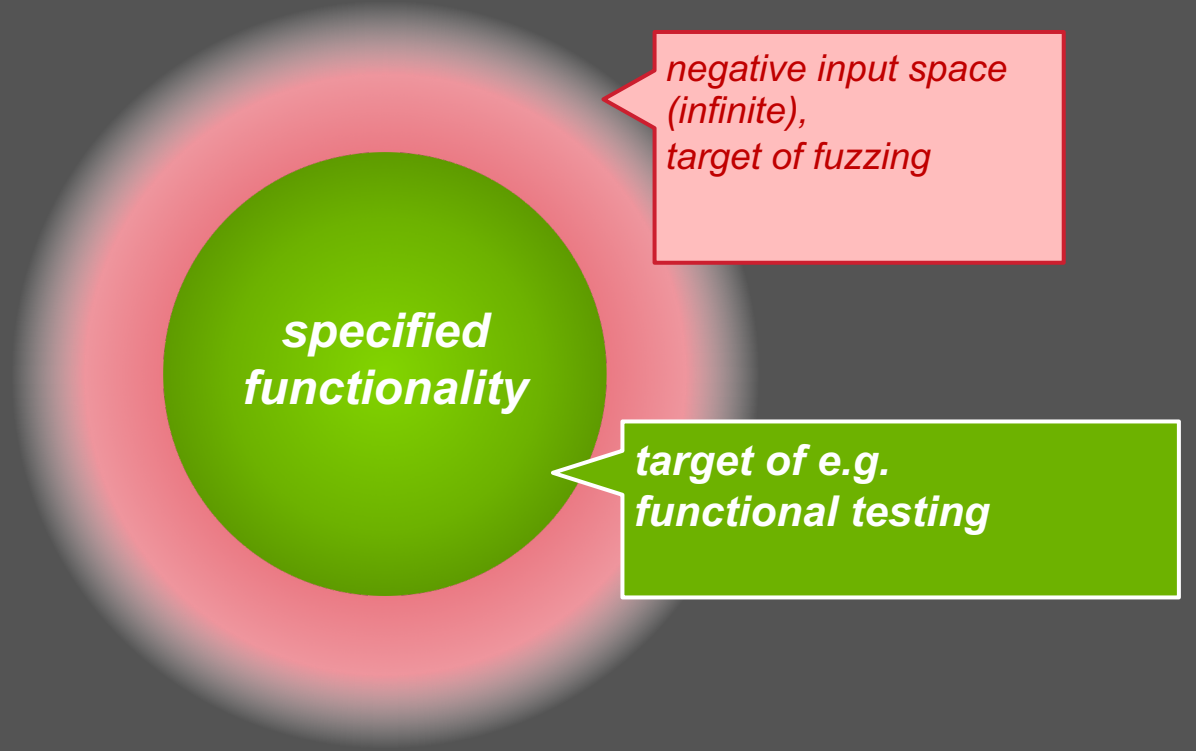
– crashes

– denial of service

– security exposures

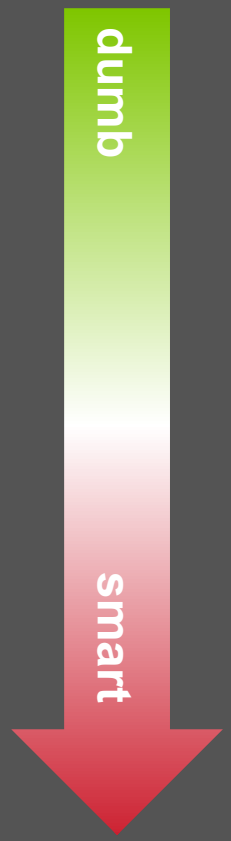
– performance degradation

highly-automated black box approach



see also: Takanen, A., DeMott, J., Miller, C.: Fuzzing for Software Security Testing and Quality Assurance. Artech House, Boston (2008)

Categorization of Fuzzers



Random-based fuzzers

– nearly no protocol knowledge

Template-based fuzzers

– use samples as starting point for mutation

Block-based fuzzers

– break messages down into static and dynamic parts an fuzz only dynamic parts

Dynamic Generation/Evolution-based fuzzers

– employ model inference algorithm to learn the test model and fuzz implicitly

Model-based fuzzers

– employ models of the input domain for generating systematic, non-random test cases

Model-Based Fuzzing

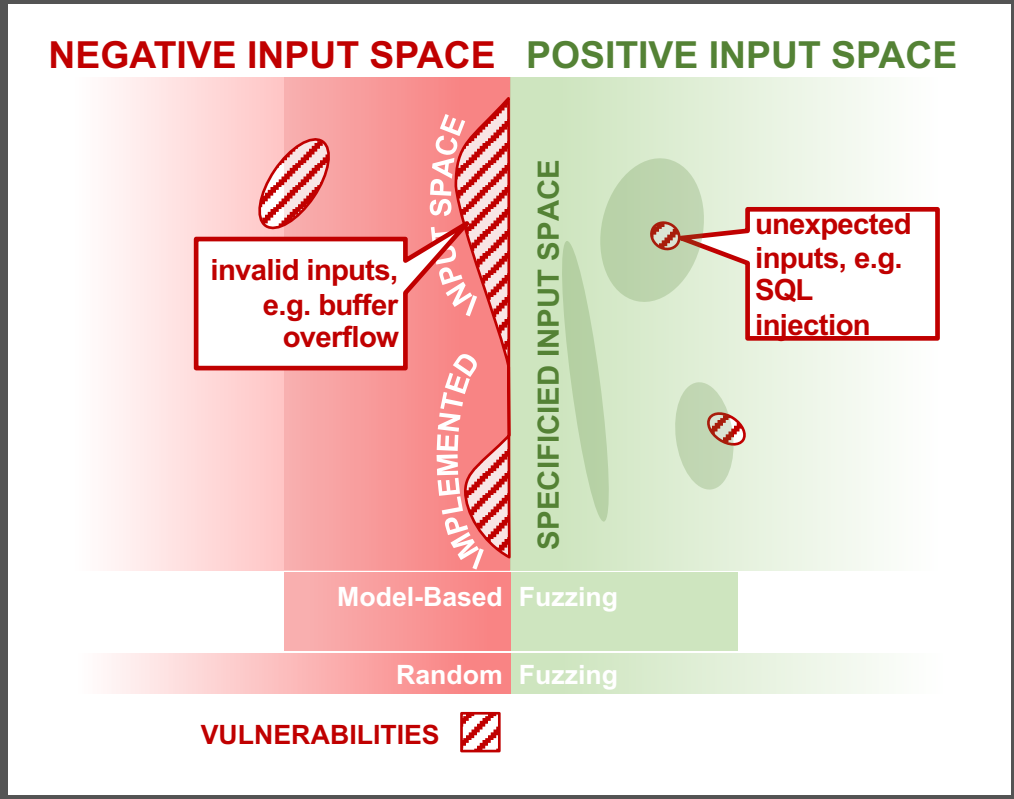
model-based fuzzers employ models of the input domain for generating systematic, non-random test cases, e.g.,

- Context-free grammars
- state machines
- stochastic models

the model is used to generate complex interaction with the SUT

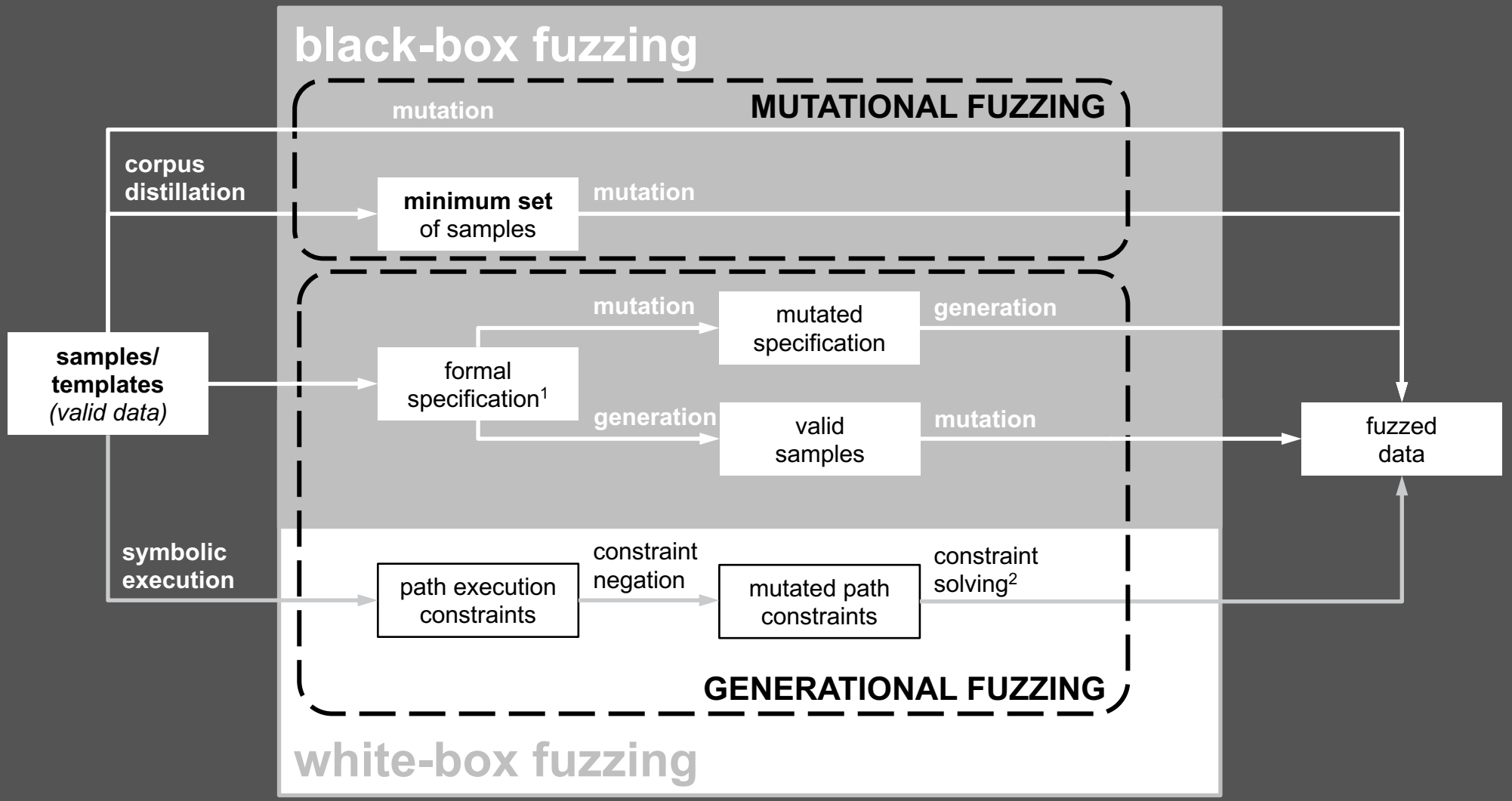
employ fuzzing heuristics to reduce the input space of invalid and unexpected inputs

model-based fuzzers finds defects which human testers would fail to find



Fraunhofer FOKUS

Fuzzing Approaches



[1] e.g., grammars, models
[2] may use a grammar additionally

Fraunhofer FOKUS

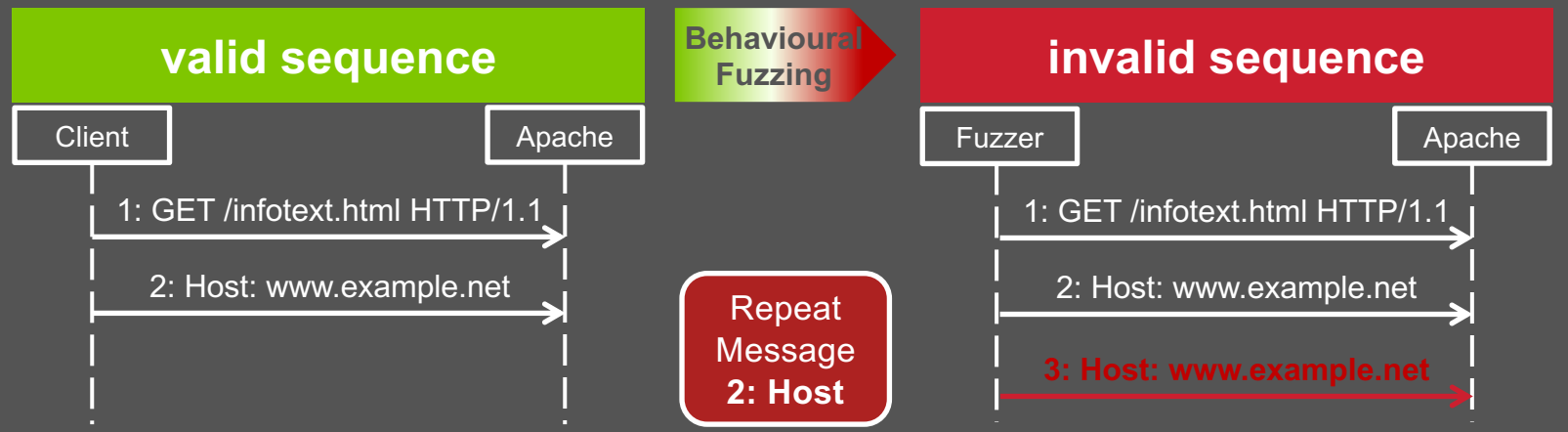
Model-Based Behavioral Fuzzing

Test cases are generated by fuzzing valid sequences, e.g., functional test cases.

Behavioral fuzzing is realized by changing the order and appearance of messages in two ways

- by rearranging messages directly
- by modifying control structures

Invalid sequences are generated by applying fuzzing operators to a valid sequence.



Fraunhofer FOKUS

Fuzz Testing: Challenges & Pitfalls

Selecting an appropriate fuzzing approach

Test automation and dealing with invalid values

- ⇨ **Coverage and stop criteria**
- ⇨ **Repeatability**
- ⇨ **Checksums, encryption/signatures, length fields**
- ⇨ **Test oracle problem**

Coverage and Stop Criteria

Coverage of previous vulnerabilities measures the occurrences of revealed vulnerabilities that have already been known before

Interface coverage indicates the number of the test item's interfaces that are targeted by the fuzzing process

Specification coverage similar to interface coverage

Input space coverage measures the volume of the potential fuzz test cases

Code coverage measures the percentage of the test item's code that has been executed while processing the fuzz test cases

Stop criteria minimum of 500,000 iterations and having at least 250,000 iterations since the last bug was found for file fuzzing (Microsoft)

There are no fixed numbers and rules!

Repeatability

crucial property for fuzzing

required for analysis, debugging purposes, retesting and regression testing.

can be achieved by

seed for RNG: execution time for a large set of test cases

complete logging: large amount of logging data

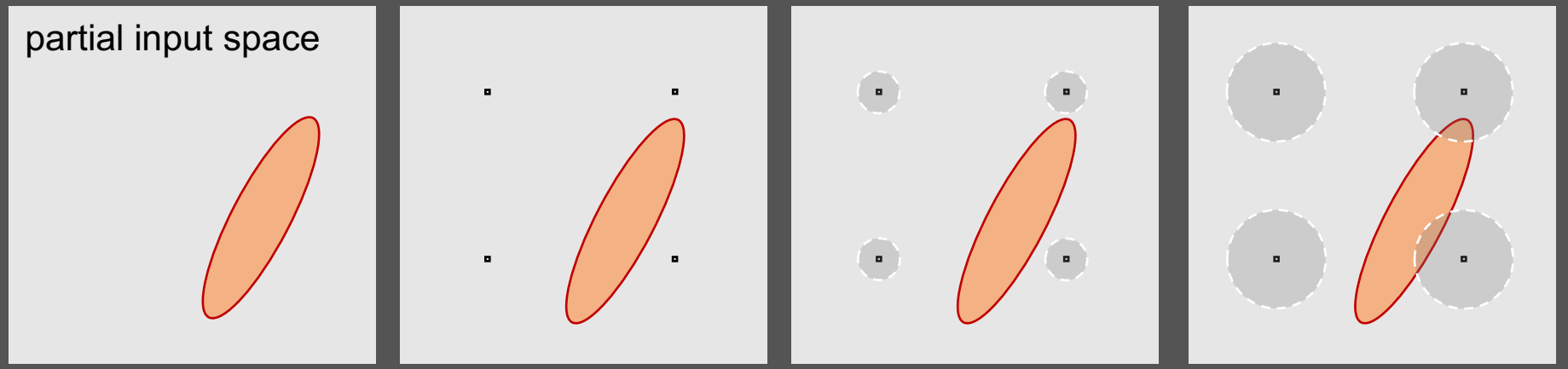
many fuzzing approaches rely to a certain degree on randomness

crashes may happen after a large bunch of test cases has been executed

race conditions

Further Challenges

Random variation



Samples

heterogeneity and diversity and uniform distribution

Checksums, encryption/signatures, length fields

Fuzzino

make traditional data fuzzing widely available

allow an **easy integration into existing tools**

without deep knowledge about fuzz data generation

allow data fuzzing without the need for

– **making familiar** with a specific fuzzing tool

– **integrating further fuzzing tools** into the test process

approach: didn't reinvent the wheel, used the potential of existing fuzzing tools (Peach, Sully)

supports regular expressions, grammars (ABNF), complex data types

basic version available on GitHub: https://github.com/fraunhoferfokus/Fuzzino/

Fraunhofer FOKUS

Test Oracle Problem

functional testing:

expected response for each stimulus

to determining the test verdict

non-functional, e.g. security testing:

specifying any exceptional system behavior is often infeasible

vulnerabilities do not necessarily show themselves via the test item's stimulated interface

verdict arbitration cannot rely solely on the immediate responses of the test item

requires additional observation mechanisms

monitoring the test item at runtime

Test Oracle Problem: Monitoring the Test Item

simple methods that assess the test item's behavior in a black-box manner,

- connectivity checks
- valid case instrumentation

white-box approaches

- instrumentation: modify the code of the test item in order to get more insights in the test item's internal state
- library interception: link against specialized memory allocation libraries
- virtualized environment: observe the interaction between the hardware, operating system and the test item
- most common approach: code coverage
  - good starting point
  - don't rely on this

Current Trends

Combination of static and dynamic security testing

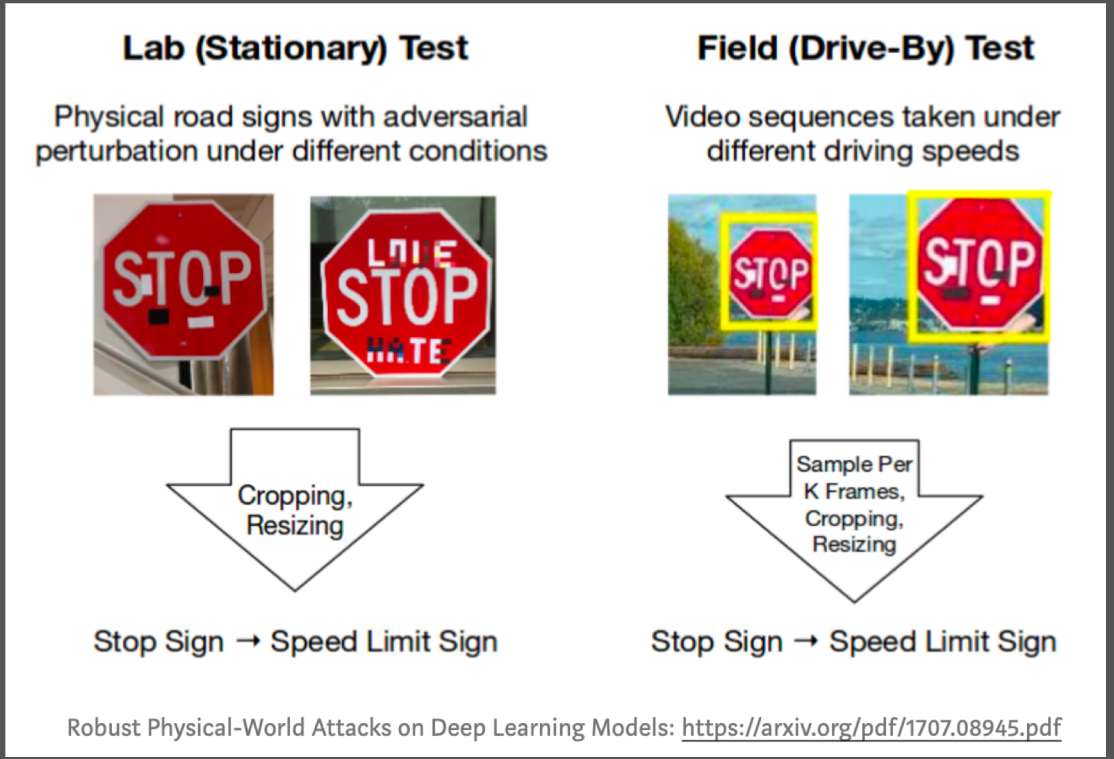
validate results from static analysis with dynamic security testing

distinguish true positives from false positives

Artificial Intelligence and Machine Learning

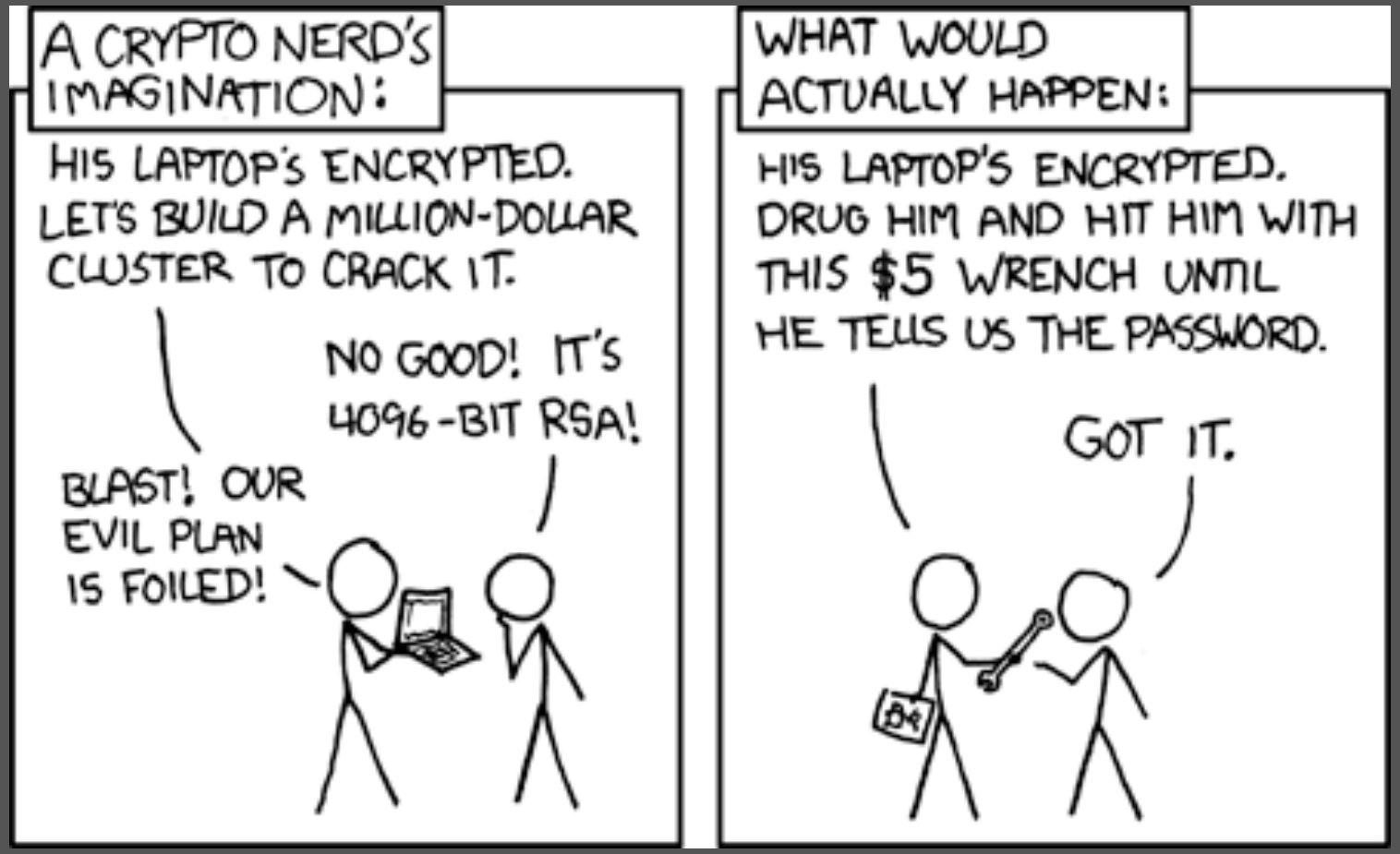
using ML for testing, e.g. for the test oracle problem

security testing for ML components and systems



Fraunhofer FOKUS

Human Factors



source: xkcd, a webcomic of romance, sarcasm, math, and language https://xkcd.com/538

Fraunhofer FOKUS

Thank you for your attention!

Fraunhofer Institute for
Open Communication Systems FOKUS
Kaiserin-Augusta-Allee 31
10589 Berlin, Germany
info@fokus.fraunhofer.de
www.fokus.fraunhofer.de

Martin A. Schneider
Phone: +49 (30) 3463-7383
martin.schneider@fokus.fraunhofer.de

Fraunhofer
FOKUS