# Week 7: Fuzz Testing

Dr Gunel Jahangirova

6CCS3SMT/7CCSMASE Software Measurement and Testing

Faculty of Natural, Mathematical & Engineering Sciences
Department of Informatics

# Fuzz Testing

Fuzzing, also known as fuzz testing, is an automated dynamic software testing technique that involves providing invalid, unexpected, or random data (fuzz) as inputs to a computer program.

**Automated**:

Input variations that humans may not think of

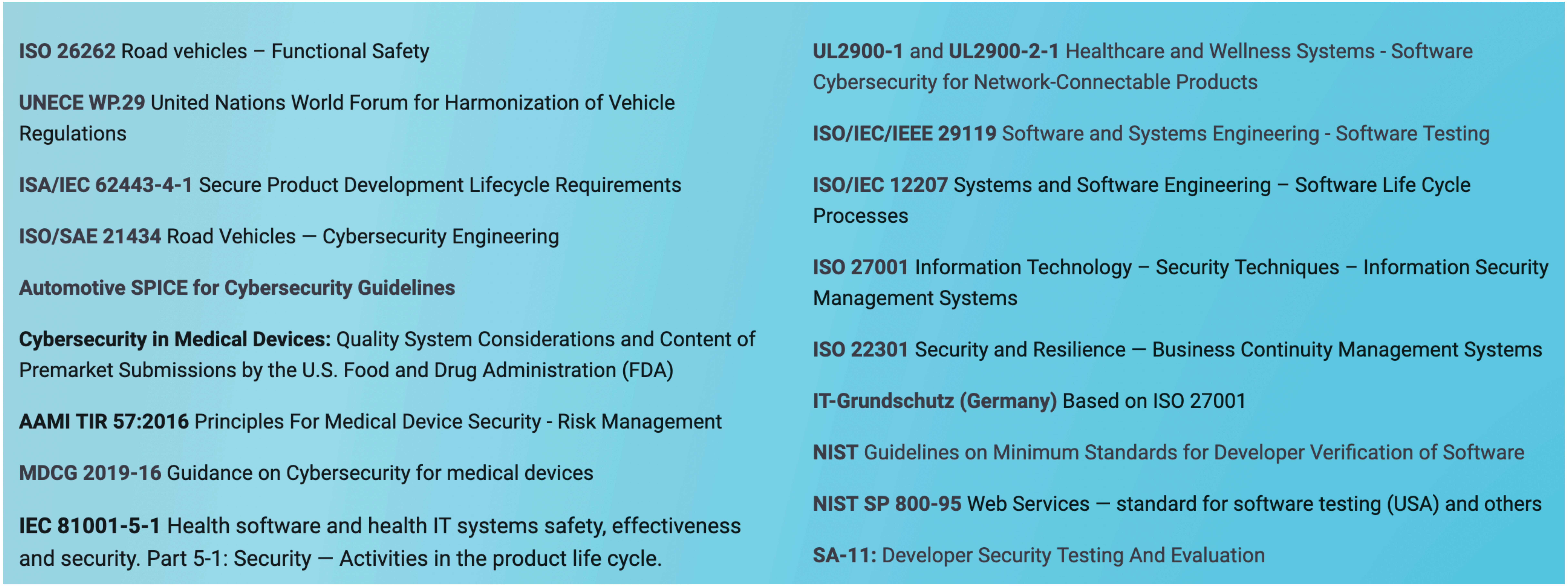
Higher number of inputs can be generated in a shorter time

**Dynamic**:

Uncovers bugs that only appear when the program is run

Takes into account the actual environment in which the software is running

# Fuzz Testing is highly recommended



**ISO 26262** Road vehicles – Functional Safety

**UNECE WP.29** United Nations World Forum for Harmonization of Vehicle Regulations

**ISA/IEC 62443-4-1** Secure Product Development Lifecycle Requirements

**ISO/SAE 21434** Road Vehicles — Cybersecurity Engineering

**Automotive SPICE for Cybersecurity Guidelines**

**Cybersecurity in Medical Devices:** Quality System Considerations and Content of Premarket Submissions by the U.S. Food and Drug Administration (FDA)

**AAMI TIR 57:2016** Principles For Medical Device Security - Risk Management

**MDCG 2019-16** Guidance on Cybersecurity for medical devices

**IEC 81001-5-1** Health software and health IT systems safety, effectiveness and security. Part 5-1: Security — Activities in the product life cycle.

**UL2900-1** and **UL2900-2-1** Healthcare and Wellness Systems - Software Cybersecurity for Network-Connectable Products

**ISO/IEC/IEEE 29119** Software and Systems Engineering - Software Testing

**ISO/IEC 12207** Systems and Software Engineering – Software Life Cycle Processes

**ISO 27001** Information Technology – Security Techniques – Information Security Management Systems

**ISO 22301** Security and Resilience — Business Continuity Management Systems

**IT-Grundschutz (Germany)** Based on ISO 27001

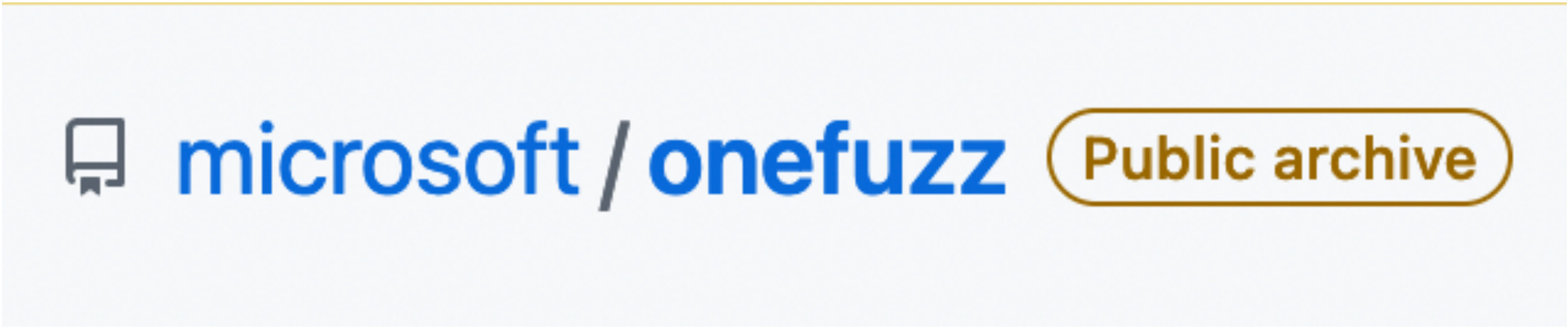
**NIST** Guidelines on Minimum Standards for Developer Verification of Software

**NIST SP 800-95** Web Services — standard for software testing (USA) and others

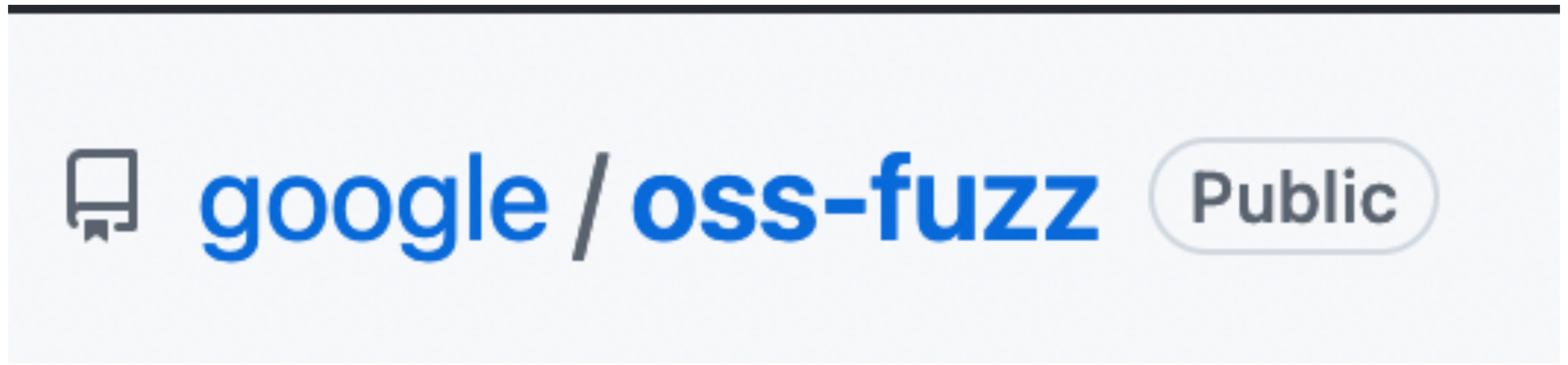
**SA-11:** Developer Security Testing And Evaluation

Source: https://www.code-intelligence.com/what-is-fuzz-testing#standards

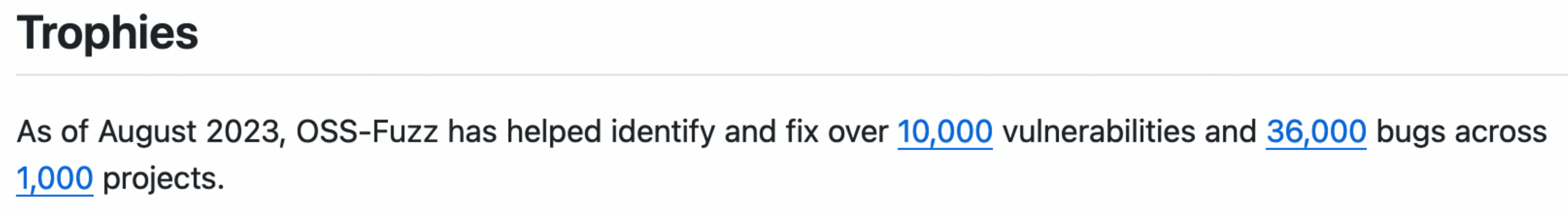
# Fuzz Testing: Industrial Adoption



microsoft / onefuzz



google / oss-fuzz



## Trophies

As of August 2023, OSS-Fuzz has helped identify and fix over 10,000 vulnerabilities and 36,000 bugs across 1,000 projects.

# Types of Fuzz Testing

By Target

By Input Structure Awareness

By Knowledge of Program Structure

By How Inputs are Generated

# Fuzz Testing: Input Structure Knowledge

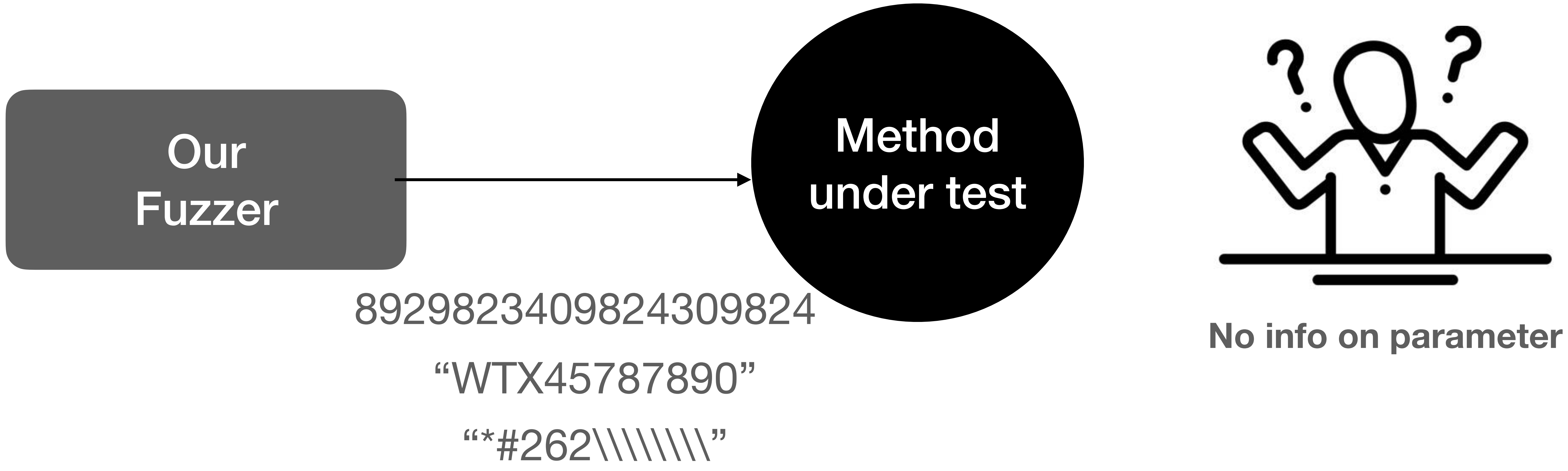
KING'S College LONDON

## Dumb (Input Structure-unaware) Fuzzing

The fuzzer has no knowledge of the input structure or format. It generates purely random inputs without any awareness of the expected input format or constraints.

## Smart (Input Structure-aware) Fuzzing

The fuzzer is aware of the structure and format of the input data. It generates inputs that conform to certain rules or syntax, making it more efficient in targeting specific areas of the program.



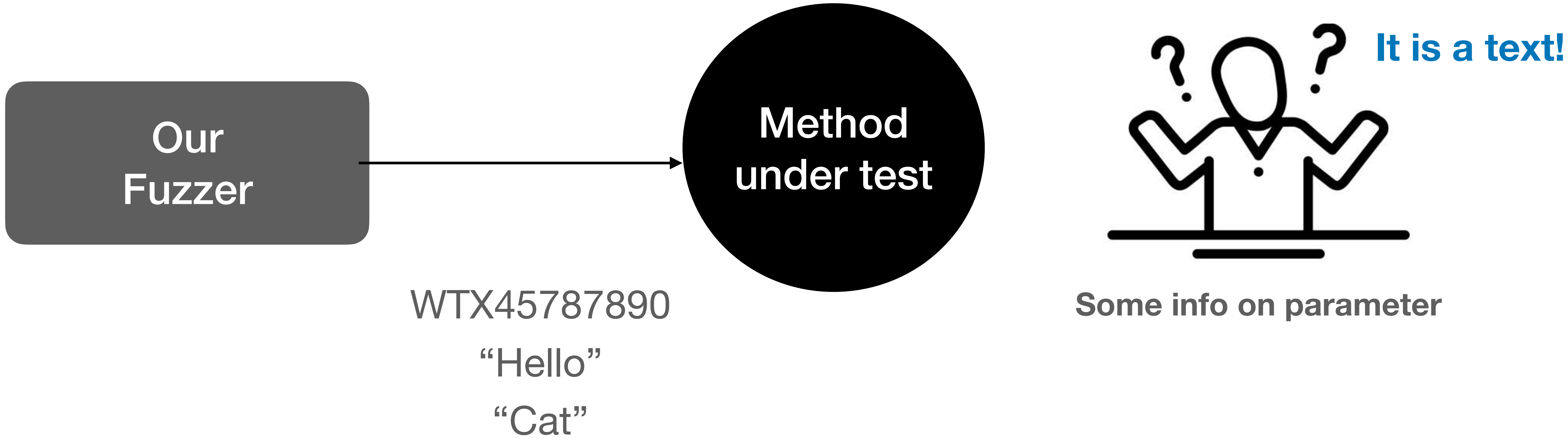
# Fuzz Testing: Input Structure Knowledge

## Dumb (Input Structure-Unaware) Fuzzing

The fuzzer has no knowledge of the input structure or format. It generates purely random inputs without any awareness of the expected input format or constraints.

## Smart (Input Structure-Aware) Fuzzing

The fuzzer is aware of the structure and format of the input data. It generates inputs that conform to certain rules or syntax, making it more efficient in targeting specific areas of the program.



WTX45787890
“Hello”
“Cat”

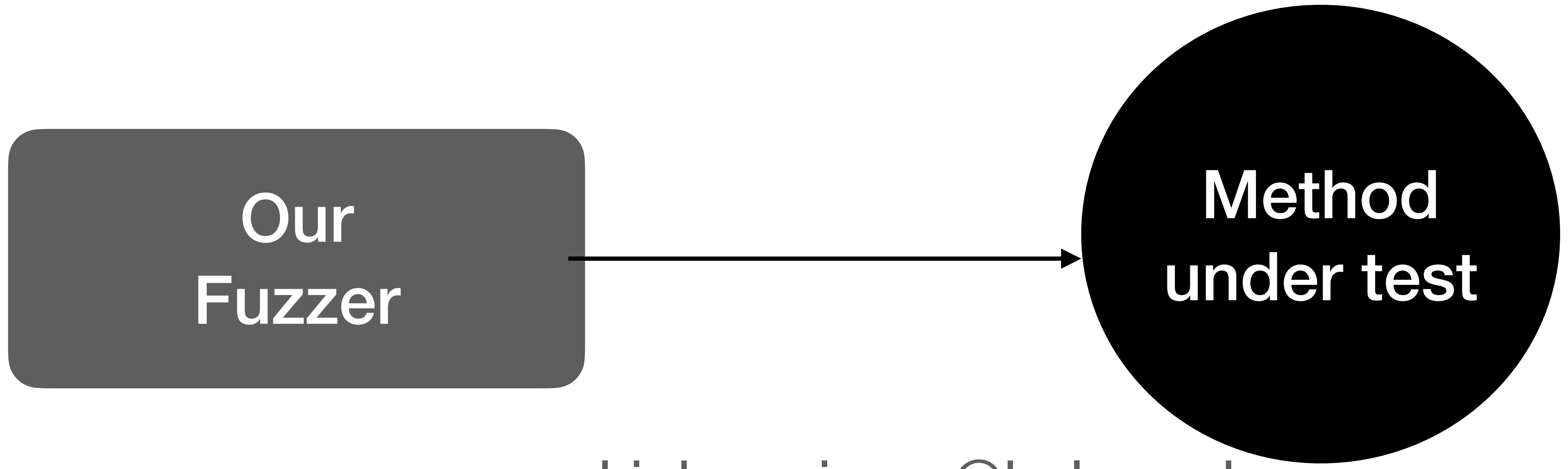
# Fuzz Testing: Input Structure Knowledge

## Dumb (Structure-unaware) Fuzzing

The fuzzer has no knowledge of the input structure or format. It generates purely random inputs without any awareness of the expected input format or constraints.

## Smart (Input Structure-Aware) Fuzzing

The fuzzer is aware of the structure and format of the input data. It generates inputs that conform to certain rules or syntax, making it more efficient in targeting specific areas of the program.



Our Fuzzer

Method under test

It is an email string!

More info on parameter

gunel.jahangirova@kcl.ac.uk

hello@gmail.com

jane@example.co.uk

# Fuzz Testing: Input Generation

### Random Fuzzing

Random fuzzing generates inputs completely at random, without any consideration for the input format or structure.

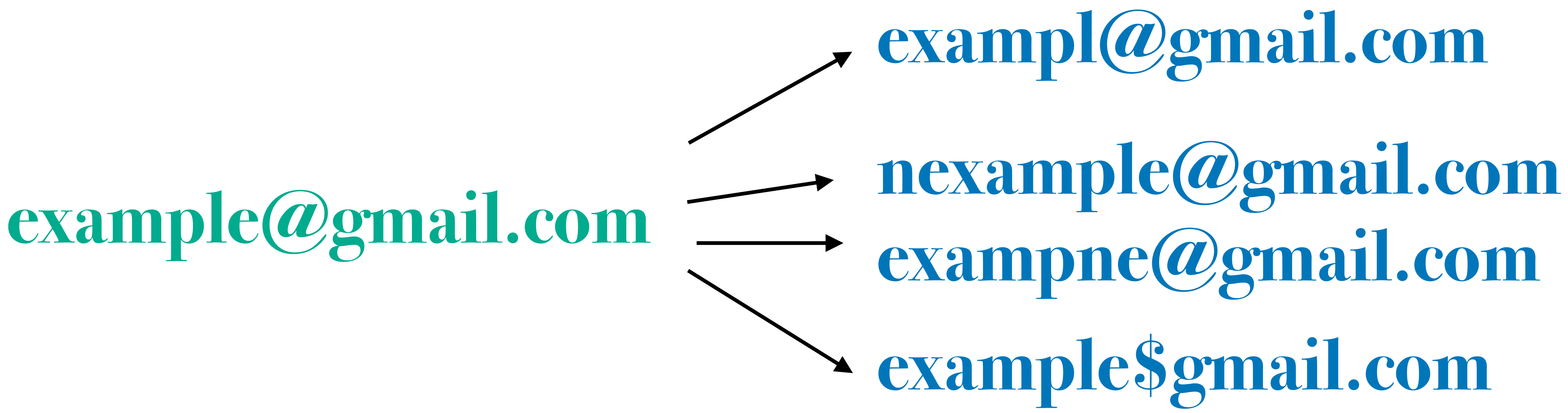
+ Simple to implement
+ Will discover basic validation issues

- Many inputs will be invalid, leading to not finding any meaningful bugs

# Fuzz Testing: Input Generation

### Mutation-Based Fuzzing

Mutation-based fuzzing begins with a set of valid input samples (seed inputs) and applies various mutations to create new test cases.



example@gmail.com

exampl@gmail.com

nexample@gmail.com

exampne@gmail.com

example$gmail.com

+ Generated inputs are more likely to be valid

+ Can uncover subtle bugs by exploring variations of known good inputs

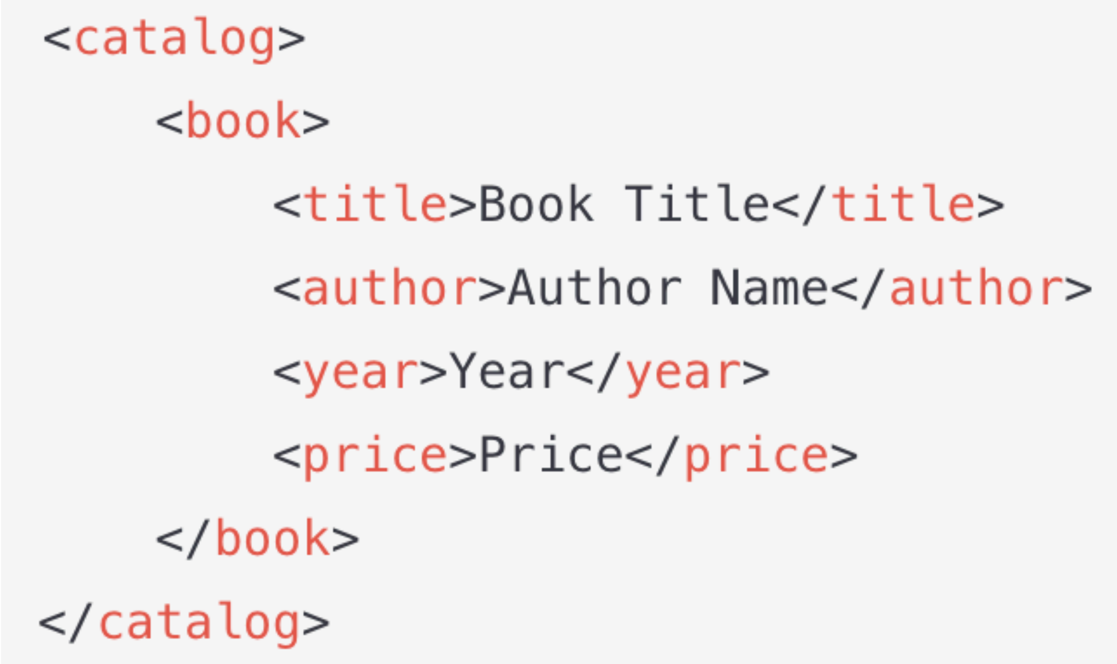
- Effectiveness relies on the quality and diversity of the seed inputs

- May not generate entirely new input types that are not represented in the seed set.

# Fuzz Testing: Input Generation

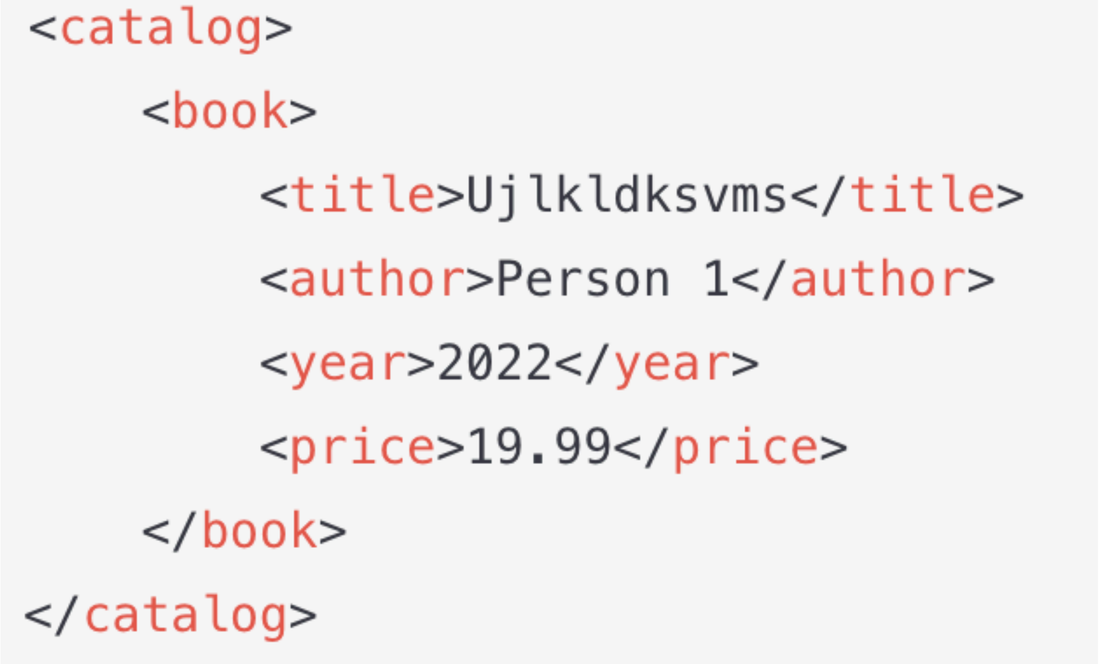
## Generation-Based Fuzzing

Generation-based fuzzing creates inputs from scratch based on a predefined set of rules or specifications that define the valid input format. This can include formal grammars or models that describe the structure of the expected input.

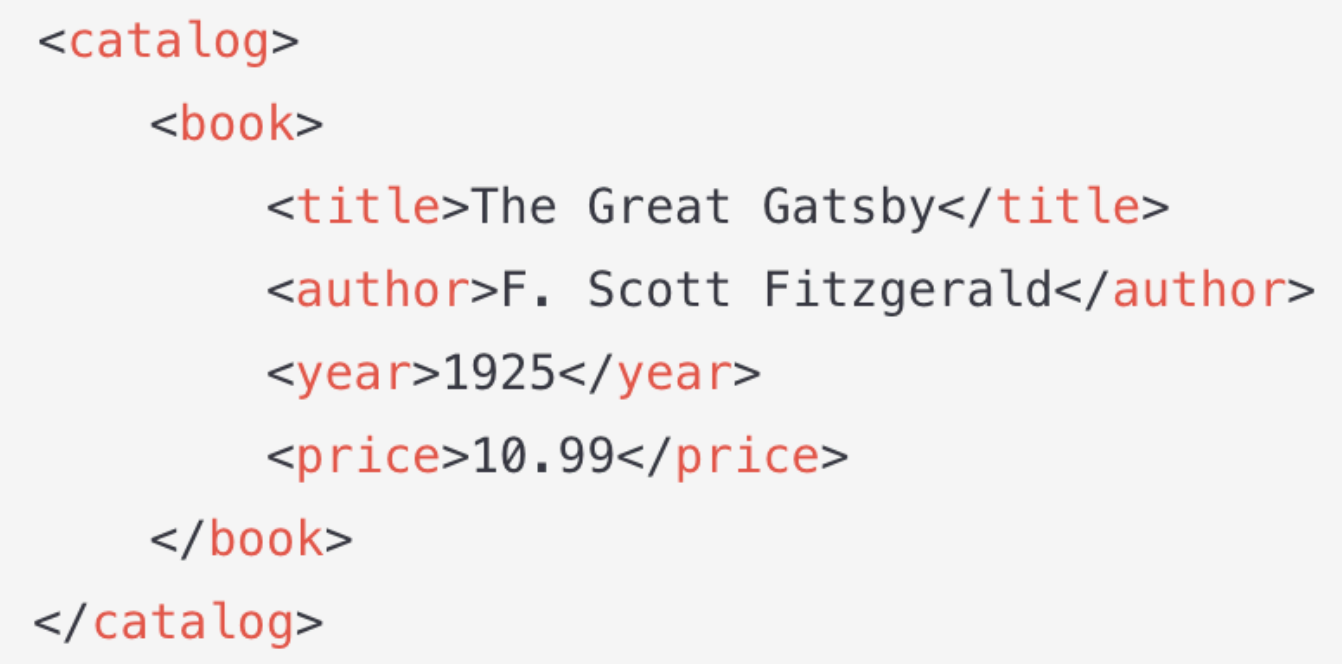


```
<catalog>
    <book>
        <title>Book Title</title>
        <author>Author Name</author>
        <year>Year</year>
        <price>Price</price>
    </book>
</catalog>
```

→



```
<catalog>
    <book>
        <title>Ujlkldksvms</title>
        <author>Person 1</author>
        <year>2022</year>
        <price>19.99</price>
    </book>
</catalog>
```



```
<catalog>
    <book>
        <title>The Great Gatsby</title>
        <author>F. Scott Fitzgerald</author>
        <year>1925</year>
        <price>10.99</price>
    </book>
</catalog>
```

+ Generates valid inputs from the outset

+ Can effectively explore complex input structures and edge cases

- Requires a thorough understanding of the input format

- May be more time-consuming to implement compared to random or mutation-based approaches

# Fuzz Testing: Program Structure

Black-box fuzz testing

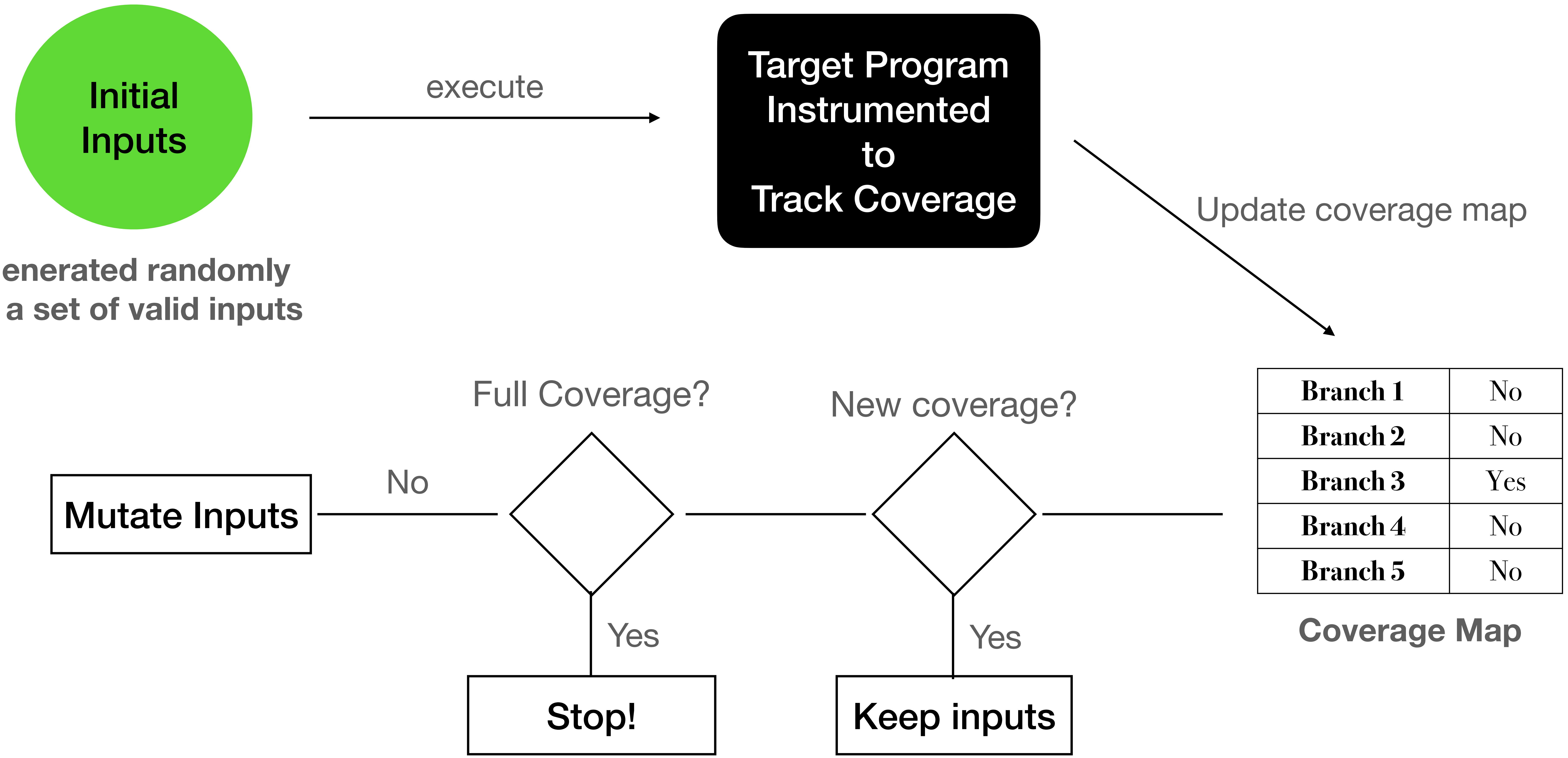
White-box fuzz testing

Gray-box fuzz testing

# Coverage-guided mutation fuzzing

Target Program

# Coverage-guided mutation fuzzing

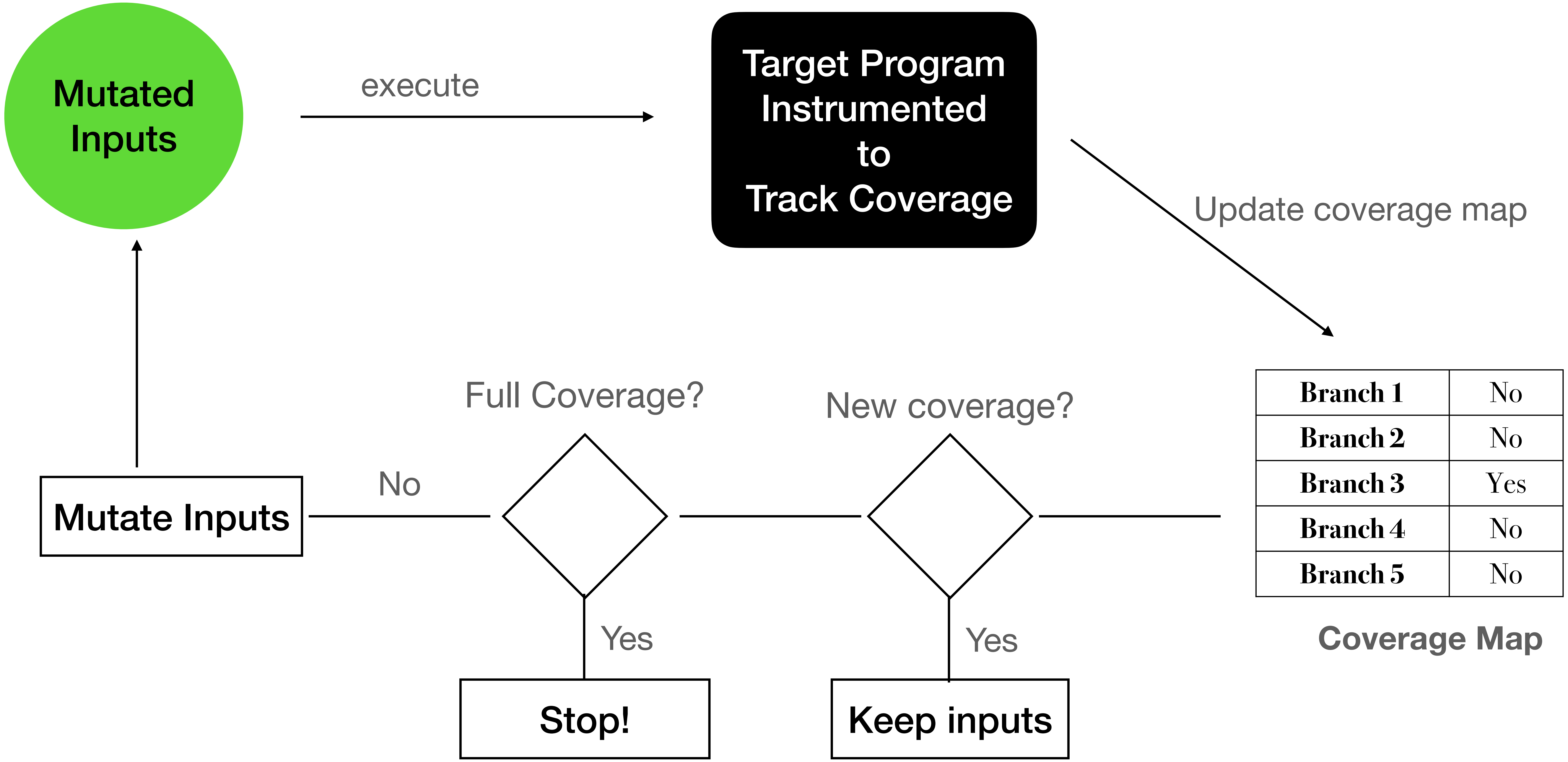


Generated randomly
or a set of valid inputs

| | |
|---|---|
| **Branch 1** | No |
| **Branch 2** | No |
| **Branch 3** | Yes |
| **Branch 4** | No |
| **Branch 5** | No |

**Coverage Map**

# Coverage-guided mutation fuzzing

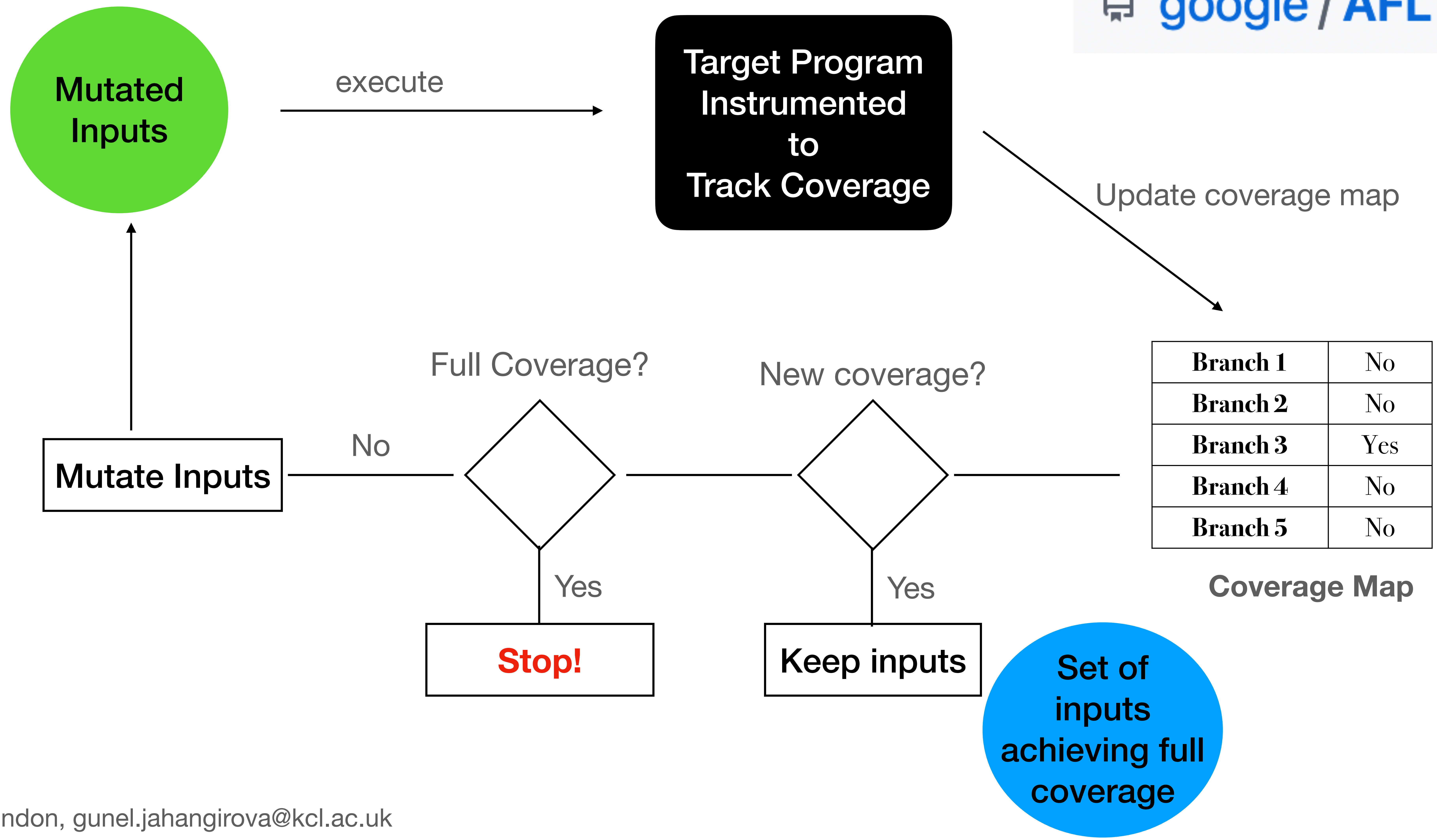


| | |
|---|---|
| Branch 1 | No |
| Branch 2 | No |
| Branch 3 | Yes |
| Branch 4 | No |
| Branch 5 | No |

**Coverage Map**

# Coverage-guided mutation fuzzing

google / AFL Public archive



| | |
|---|---|
| Branch 1 | No |
| Branch 2 | No |
| Branch 3 | Yes |
| Branch 4 | No |
| Branch 5 | No |

Coverage Map

# Fuzzing Tools

| Programming Language | Fuzzing Tools |
|---|---|
| C/C++ | AFL, LibFuzzer |
| Java | Jazzer |
| Python | Atheris |
| JavaScript | Fuzzilli |
| .NET | SharpFuzz |
| Web Applications | WFuzz |