GFDG: A Genetic Fuzzing Method for the Controller Area Network Protocol

Miguel Stey[1], Murad Hachani[1], Philipp Fuxen1,
Julian Graf[1], Rudolf Hackenberg[1]

[1]*Department of Computer Science and Mathematics*
Ostbayerische Technische Hochschule Regensburg

Contact email: miguel1.stey@st.oth-regensburg.de

OTH REGENSBURG

IARIA

Miguel Stey

Our aim:
1. A feedbackbased Fuzz Data Generator that uses the genetic algorithm and sidechannel data for the CAN protocol

2. Evaluate this method on a real ECU

Contributions:

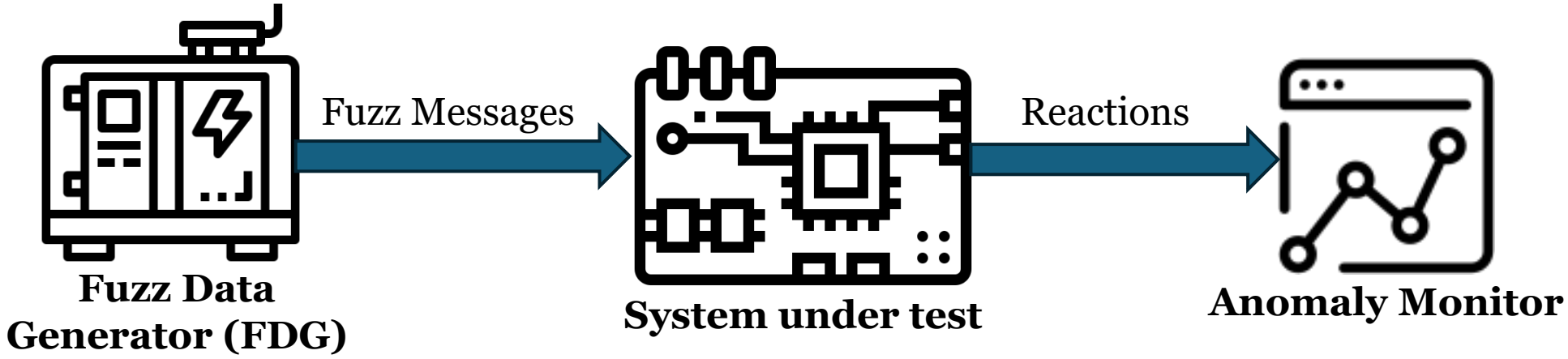
1. Identifying that sidechannel information can be used to identify reactions of a system to CAN messages
2. A promising algorithm that is able to use sidechannel information to improve the fuzz data generation

Aims and contributions of our paper

Our aim:
Develop a feedback based method for CAN fuzzing using the Genetic Algorithm.

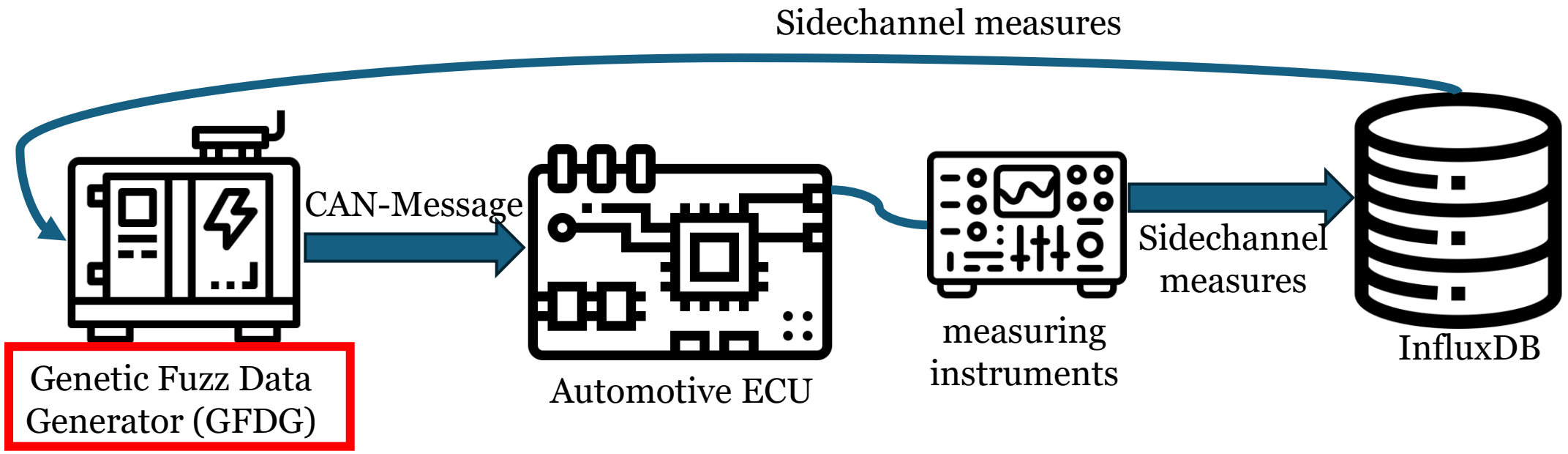
Before leaving the scientific field his research focused on the topics Cyber Security, Fuzz Testing (especially for the CAN protocol) and Enterprise Architecture.

Fuzz Testing in general



Every Fuzzer has two key components. A Fuzz Data Generator that uses a specific strategy to generate input messages to the system under test and an Anomaly Monitor that observes the reactions of the system to identify anomal behavior to identify vulnerabilities[6]

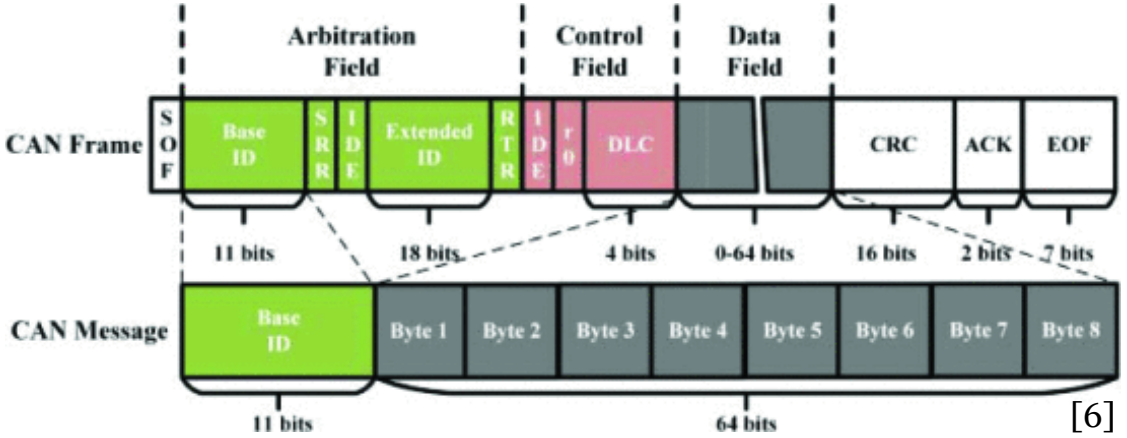
Scenario of this paper



Goal: A FDG based on the Genetic Algorithm using sidechannel measurements to improve fuzz message generation
=>Feedbackbased
(Anomaly Monitor not in scope)

Defining a CAN Individual

From the point of view of a FDG a CAN message consists of a ID and a payload.



[6]

CAN-ID: 11-Bit ID that has two main usages:

-Arrbitration

-Context of the payload

⇒ CAN ID is used by the ECU to identify which messages on the bus are relevant. If an ID is not in the list of messages relevant for the ECU it will not process it.

⇒ **ECU will only process messages with certain Ids** (unknown to the FDG in blackbox scenarios).

⇒ Large amount of Fuzz Messages will never be processed by the target ECU.

⇒ GFDG: **Identify recognized IDs and generate fuzz messages focusing on these IDs.**

<u>CAN Individual in context of the Genetic Algorithm</u>:

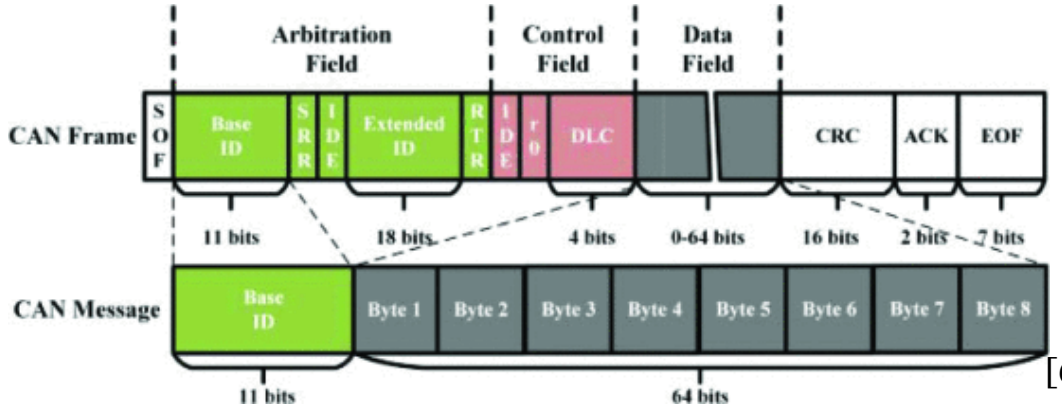
ID is a chromosom consiting of 11 genes.

After Identifying a active ID any change would lead to loss of progress.

=> This chromsom is mutation resistant and is not split during crossover.

Defining a CAN Individual

From the point of view of a FDG a CAN message consists of a ID and a payload.



[6]

Payload: 0-64 Bit Data Field

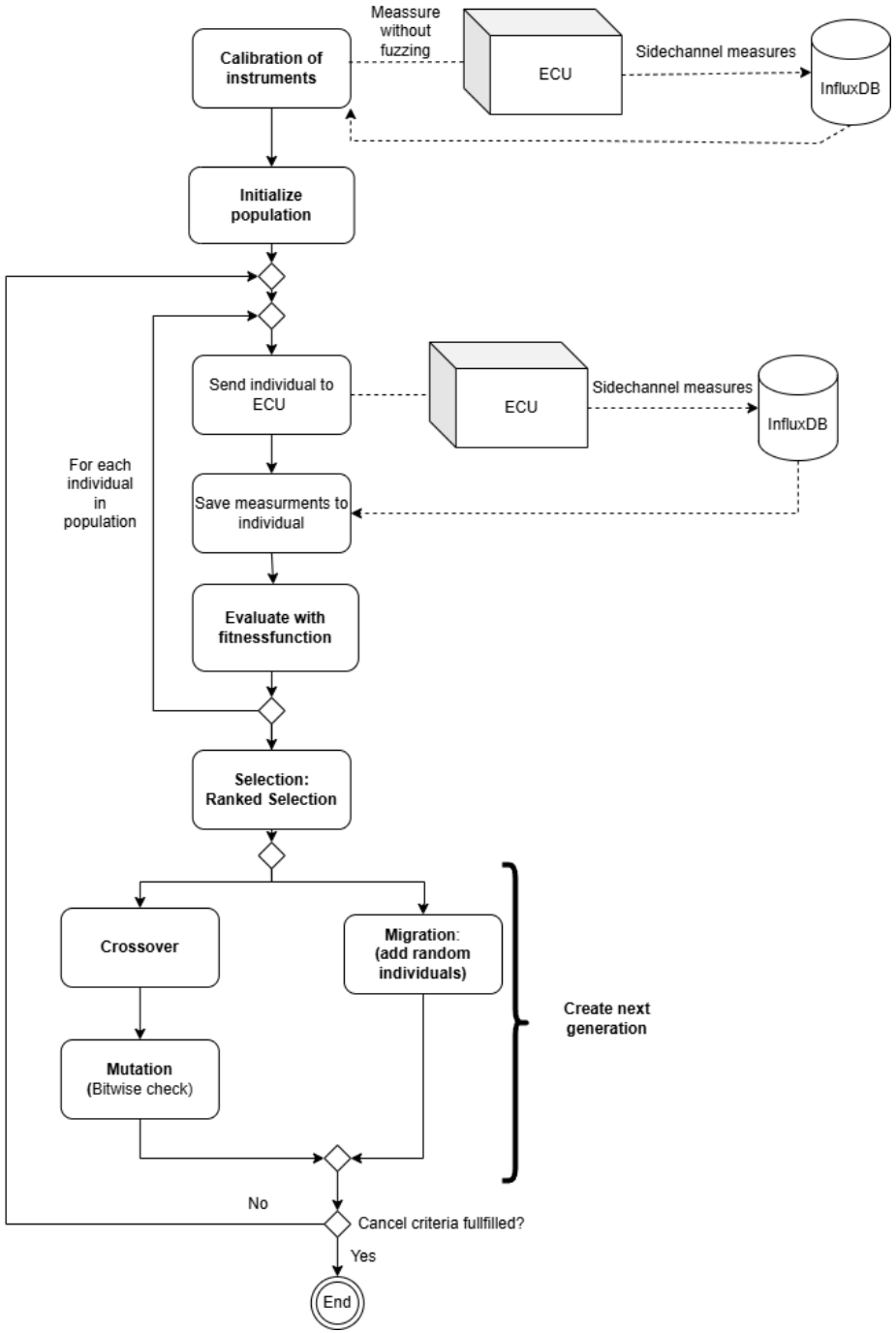
Usage of the data depends on the context of the message

(unknown to the FDG in blackbox scenarios)

<u>CAN Individual in context of the Genetic Algorithm</u>:

Payload is chromosom consiting of 0 to 64 genes

These genes are mutated by a mutation chance and are the target of the crossover step



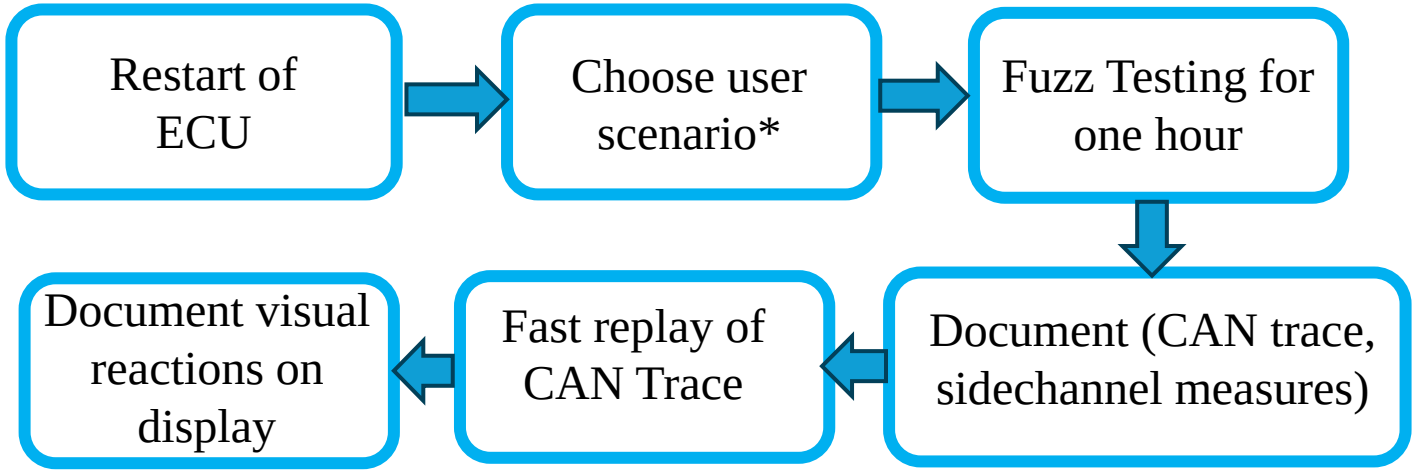
The GFDG Algorithm

Evaluation

Research Question:

How does the integration of genetic algorithms with side-channel feedback improve the identification of active CAN IDs?

Experiments: Infotainment System



***Set of Scenarios:**

-Radio Menu opened

-Main Menu opened

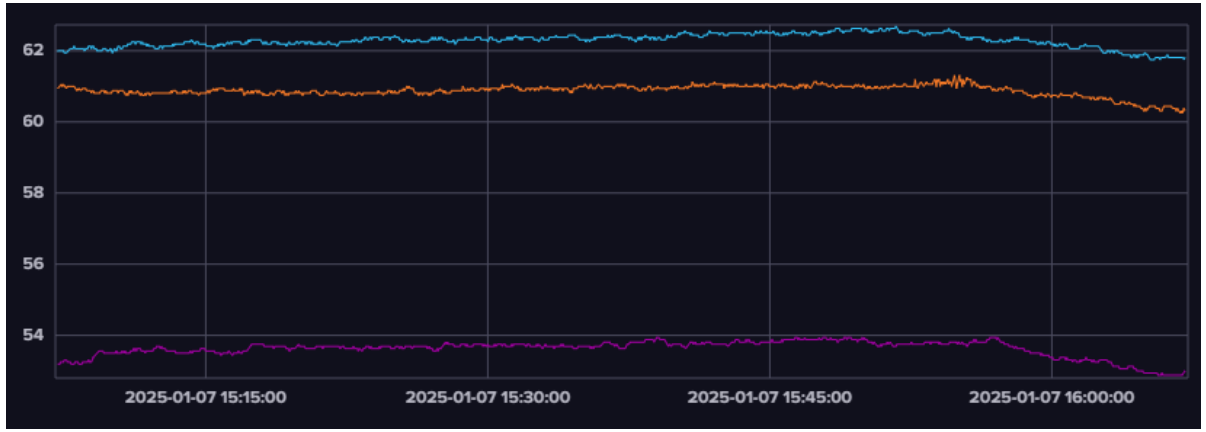
-Media Menu opened

-Navigation Menu opend

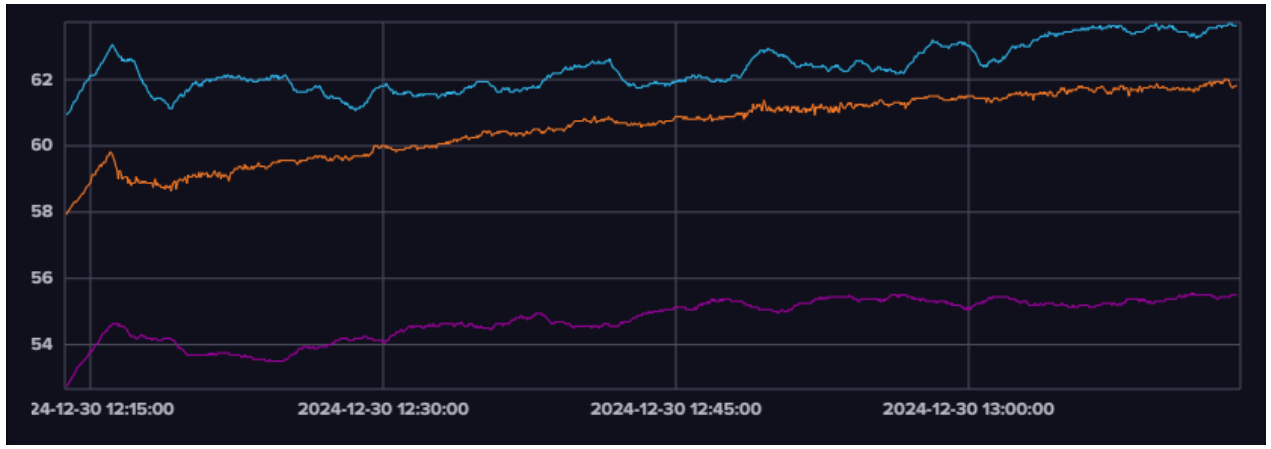
+All of the above but with connected mobile phone via bluetooth

Differences in measures

Temperature without Fuzz Testing



Temperature with fuzz testing



⇒ **For every measurement run without fuzz testing ECU sidechannels behaved consistently**
⇒ **For most (80%) runs with fuzz testing irregular behaviors on temperature sidechannel can be seen**

(Note: consistent behavior during fuzz testing is expected to a certain degree:
because certain runs have to exist where no active IDs are generated and therefore no major reactions should be measured)

temperature sensor
- CPU
- GPU
- AMC

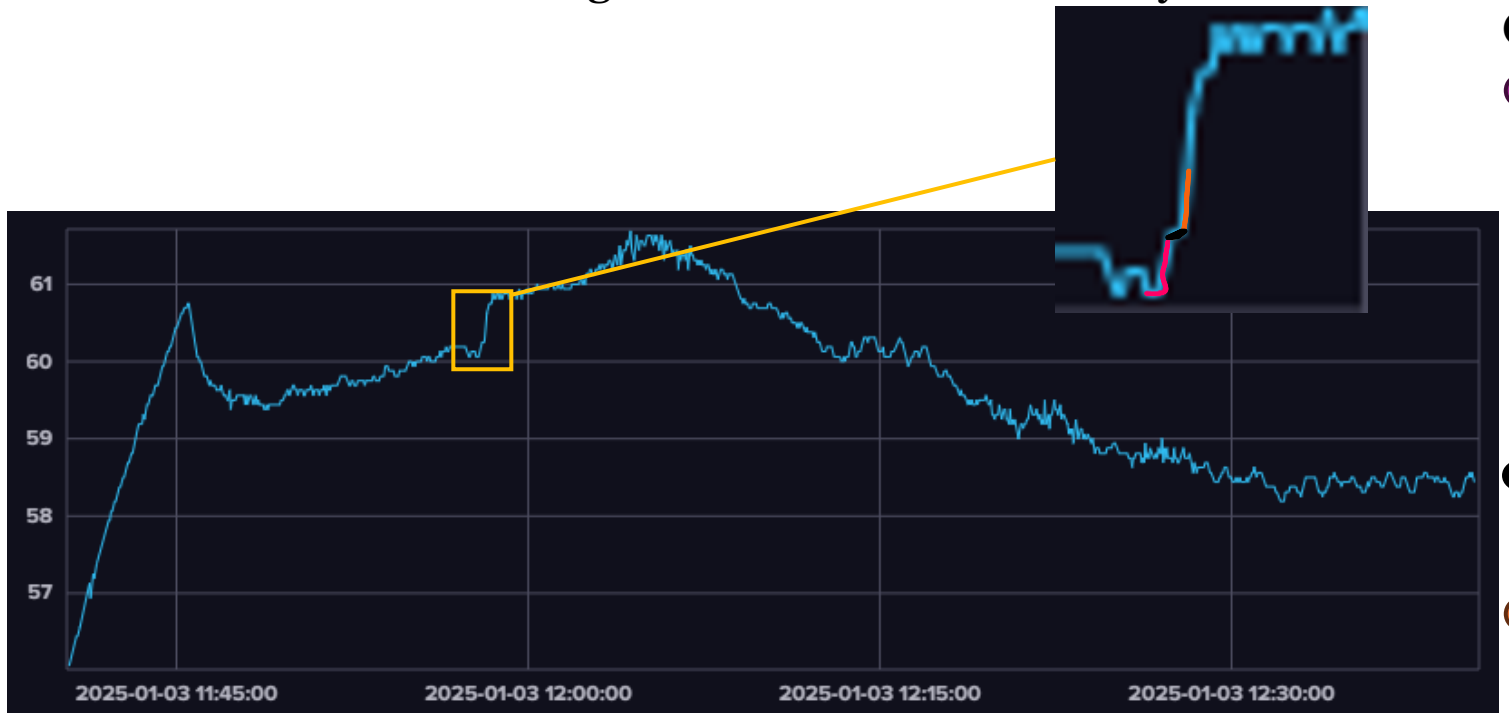
More examples of fuzzing measure

temperatur sensors

- CPU
- GPU
- AMC

Identified Bluetooth DoS

Ideal scenario:
GFDG **generates** a „**harmless**“ message **with ID X** that causes an expected reaction
➔ Fitnessfunction **recognizes this** & **generate** messages **with ID X but different payload**
➔ One of the child messages identifies a vulnerability



GFDG generated message by migration:

- 3C0#99e5316247e565
  ⇒ increase in GPU temperature
  ⇒ Fitness score increases
  ⇒ 3 children werden gezeugt mit ID **3C0**

  3c0#984b616a3827008b
  3c0#99e531604f772e8f
  3c0#99e50290c0d05e10

- First messages of new generation are sent (no reaction)
- Child message send: 3C0#99e50290c0d05e10

This behavior would be recognized by Anomalie Monitor normally, because we have none yet we need to check manually

Manual investigation

Manual investigation:

- Identifing the messages
- Replay these messages
- Monitor the behaviours

⇒ **DoS example:**

1. **First 3C0 message** triggers „**Goodbye Message**“ -> **no Probleme but increased GPU temperature**
2. First two 3C0 children: **no** reaction of system
3. **Third 3C0 Kind reboots bluetooth &** reconnects last connected device

=> repeated send (2-3x within2 seconds)

->**crash of multiple parts of the Infotainments System**

- No access to Radio, Media & Telephone menus of the Infotainment System
- No changes to bluetooth settings possible
- Only fix: complete reboot of the Infotainment Systems

Conclusion & future work

- Reactions to fuzz messages recognizable in sidechannel measure
- Sidechannel feedback can be used in combination with the Genetic Algorithm to generate new messages for identified IDs (see DoS example)
- **Limitation**: Strong **dependency** on chosen sidechannels

⇒ GFDG reasonable for CAN-Fuzzing,
as long as expressive sidechannels are available

Possible future work: Further investigation on sidechannel usage in fuzz testing (especially anomalie detection)