DESIGN AUTOMATION CONFERENCE

FROM CHIPS TO SYSTEMS — LEARN TODAY, CREATE TOMORROW

DEC 5 - 9, 2021 ◆ San Francisco, California

PAVFuzz: State-Sensitive Fuzz Testing of Protocols in Autonomous Vehicles

Feilong Zuo, Zhengxiong Luo, Junze Yu, Zhe Liu, Yu Jiang

zuofl19@mails.tsinghua.edu.cn

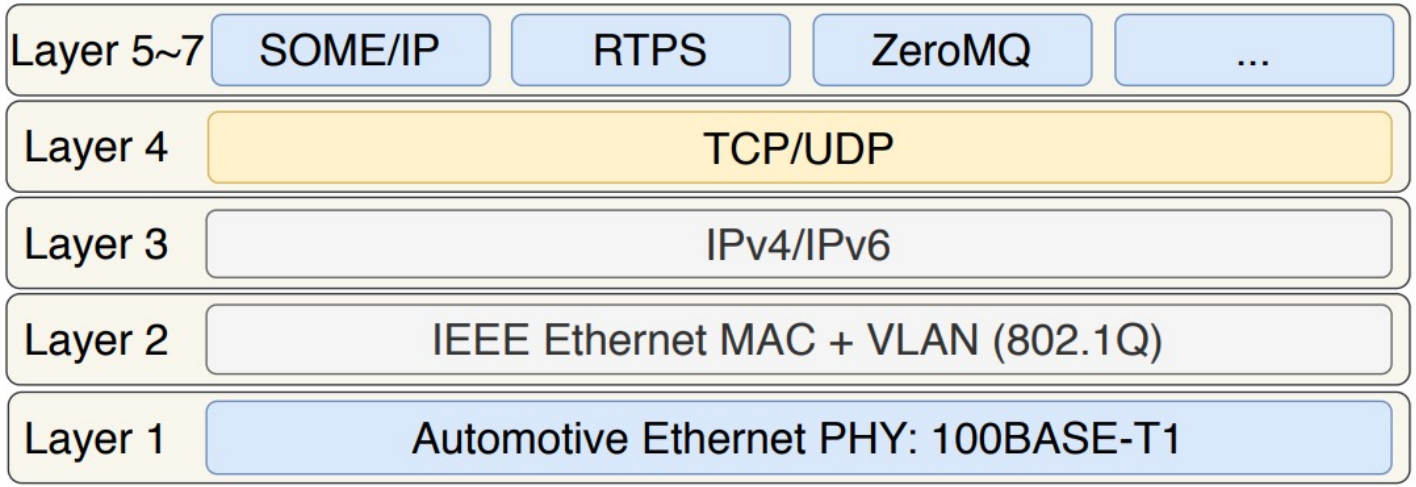
Security In Vehicular Network Systems

In July 2015, Charlie Miller and Chris Valasek once successfully injected into a Jeep Cherokee Car with the Uconnect System via the access of remote network .

They took over the right of the entertainment system, the power system, etc, while the driver sitting in the car could not give any effective orders.

It is urgently needed to guarantee the security of in-vehicle network!

Network Protocols Used in Autonomous Vehicles



A sample protocol stack in Ethernet-based autonomous vehicles:

- **Physical Layer:** 100BASE-T1 Ethernet
- **Link Layer:** MAC, VLAN
- **IP Layer:** IP protocols, including IPv4, IPv6
- **Transport Layer:** TCP/UDP protocols
- **Application Layer:** SOME/IP, RTPS, ZeroMQ ...

Generation-based Fuzz Testing of Protocols

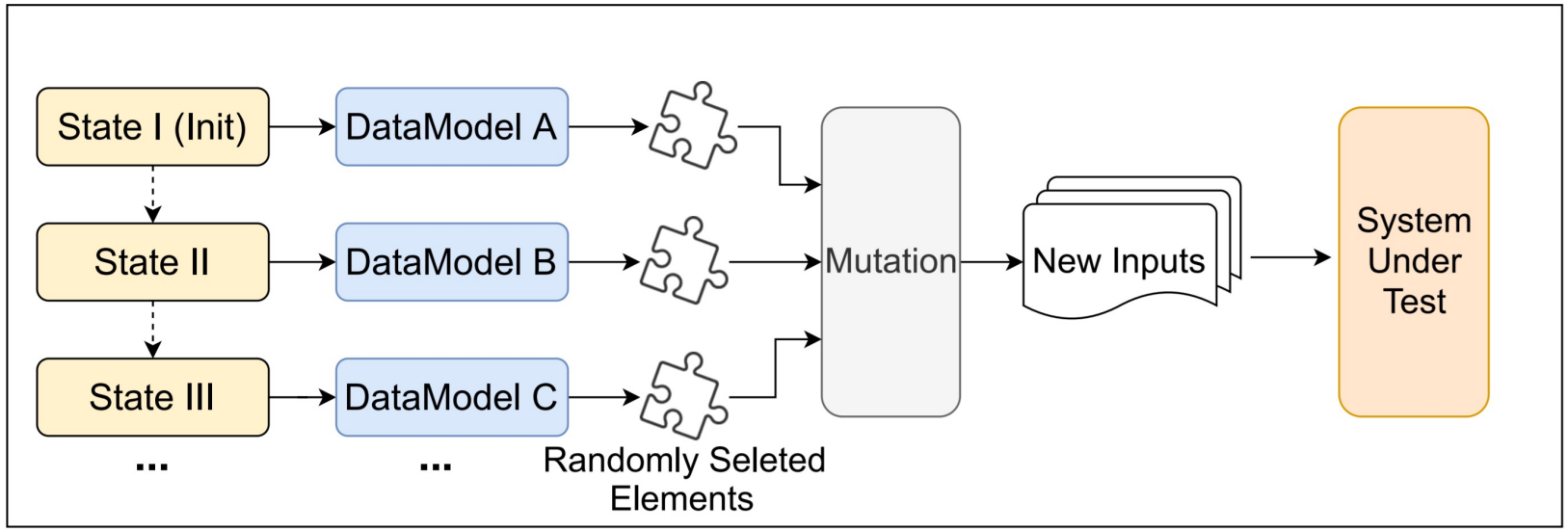


Fig. Workflow of generation-based fuzzing strategies.

- Fuzzers usually produce large number of mutated inputs to the SUT to find potential bugs
- Generation-based fuzzers are more suitable for fuzzing protocols due to the highly structured packets
- For each data model, they randomly selected several data elements and mutate them

Motivation

Relations between protocol states: RTPS as an example

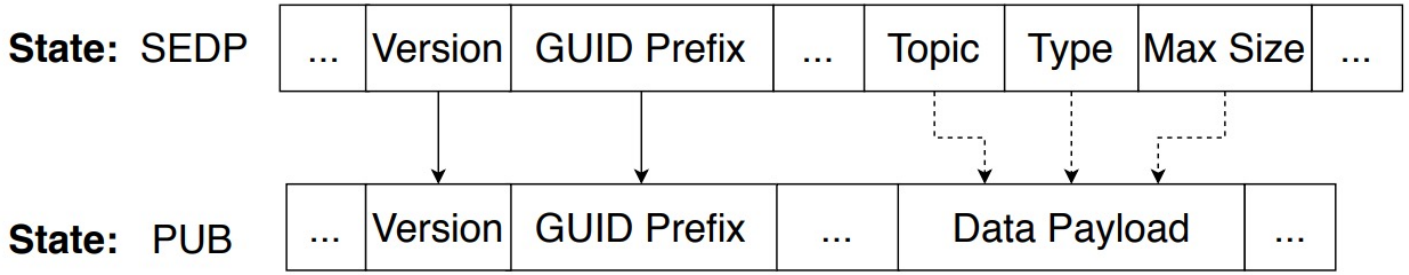


Fig. Sample relations between data elements in different RTPS states.

- **Version -> Version, GUID Prefix -> GUID Prefix**
  - Their values should keep the same between states
  - Otherwise, the under-test protocol will directly reject the following packet due to the inconsistence

Motivation

Relations between protocol states: RTPS as an example

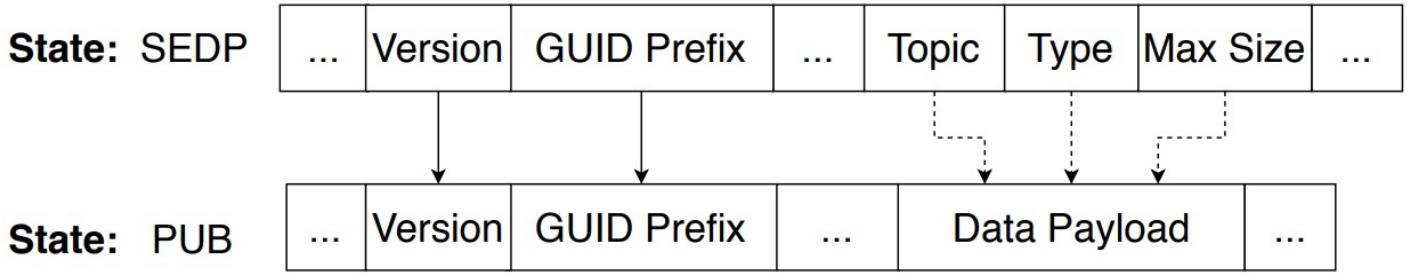


Fig. Sample relations between data elements in different RTPS states.

- **Version -> Version, GUID Prefix -> GUID Prefix**
  - Their values should keep the same between states
  - Otherwise, the under-test protocol will directly reject the following packet due to the inconsistence
- **Topic, Type, Max Size -> Data Payload**
  - These previous elements describe how the following elements are processed by protocol
  - The mutation probability of the following elements should be increased if the previous ones change

Motivation

Relations between protocol states: RTPS as an example

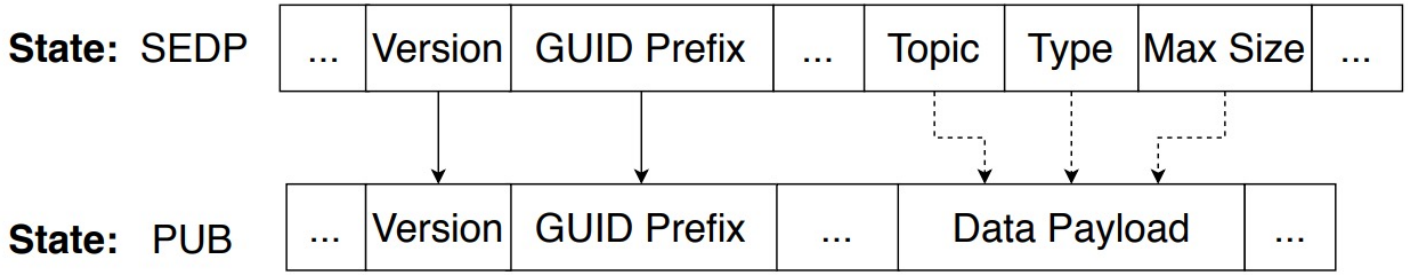


Fig. Sample relations between data elements in different RTPS states.

- **Version -> Version, GUID Prefix -> GUID Prefix**
  - Their values should keep the same between states
  - Otherwise, the under-test protocol will directly reject the following packet due to the inconsistence
- **Topic, Type, Max Size -> Data Payload**
  - These previous elements describe how the following elements are processed by protocol
  - The mutation probability of the following elements should be increased if the previous ones change
- **Other Complex Relations in Protocols**

Motivation

Traditional protocol fuzzing strategies: unable to learn and leverage these relations

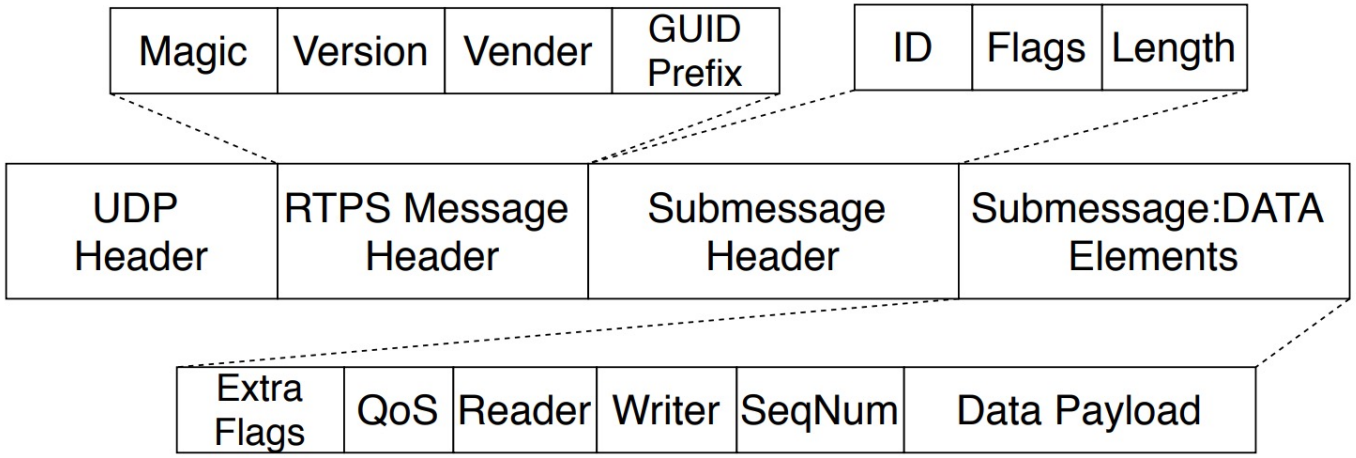
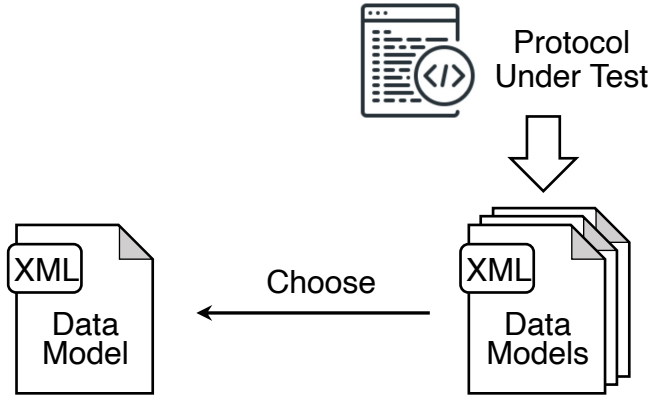


Fig. Basic template packet structure employed by RTPS protocol.

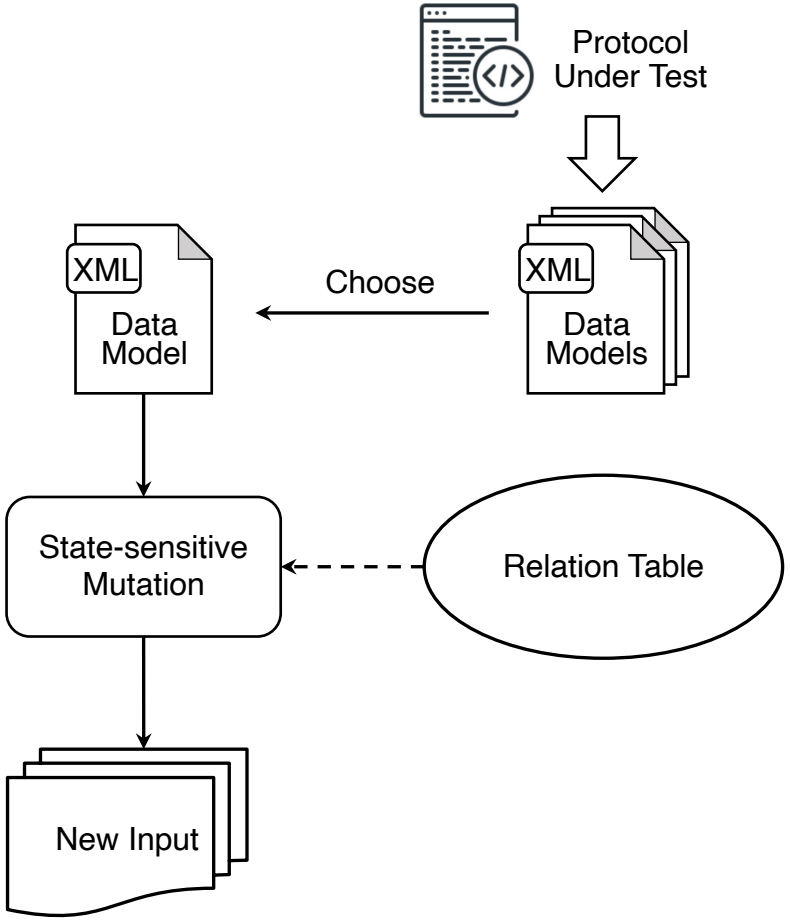
- Complex structure with many specific data elements
- Without relations, only able to **randomly** select several elements to mutate in each state
- Low fuzzing efficiency and poor effectiveness

Overview of PAVFuzz



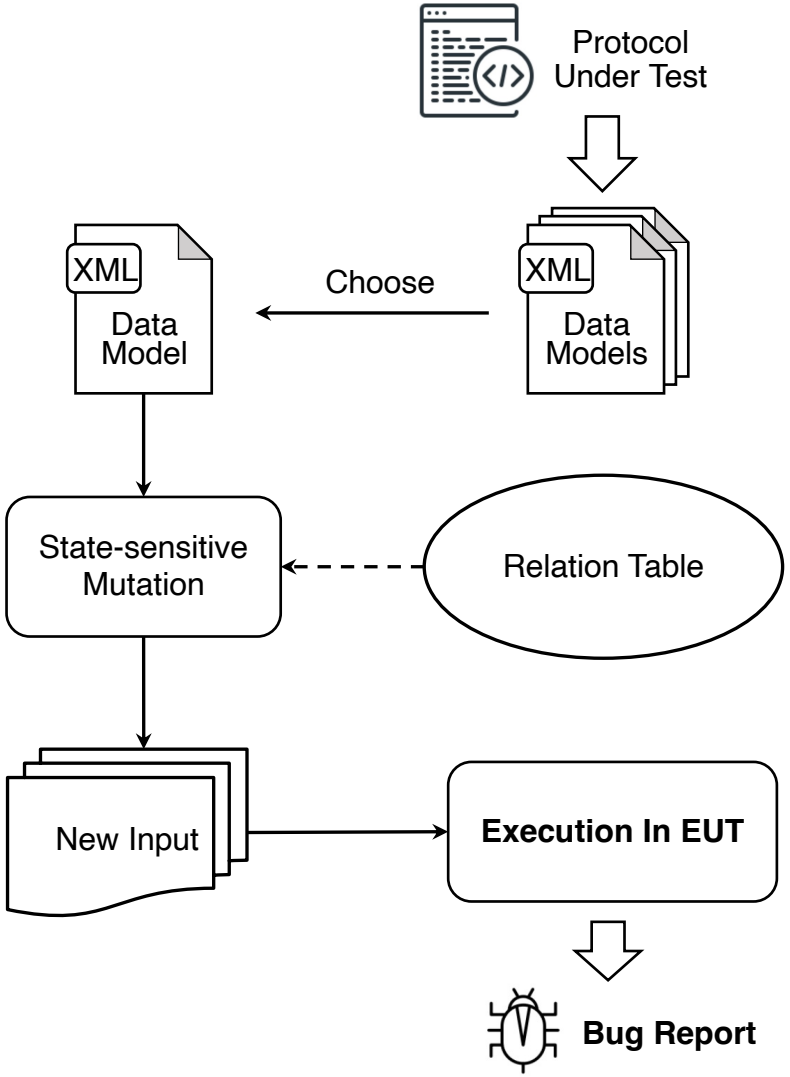
- PAVFuzz starts with user-provided data models of the under-test protocol

Overview of PAVFuzz



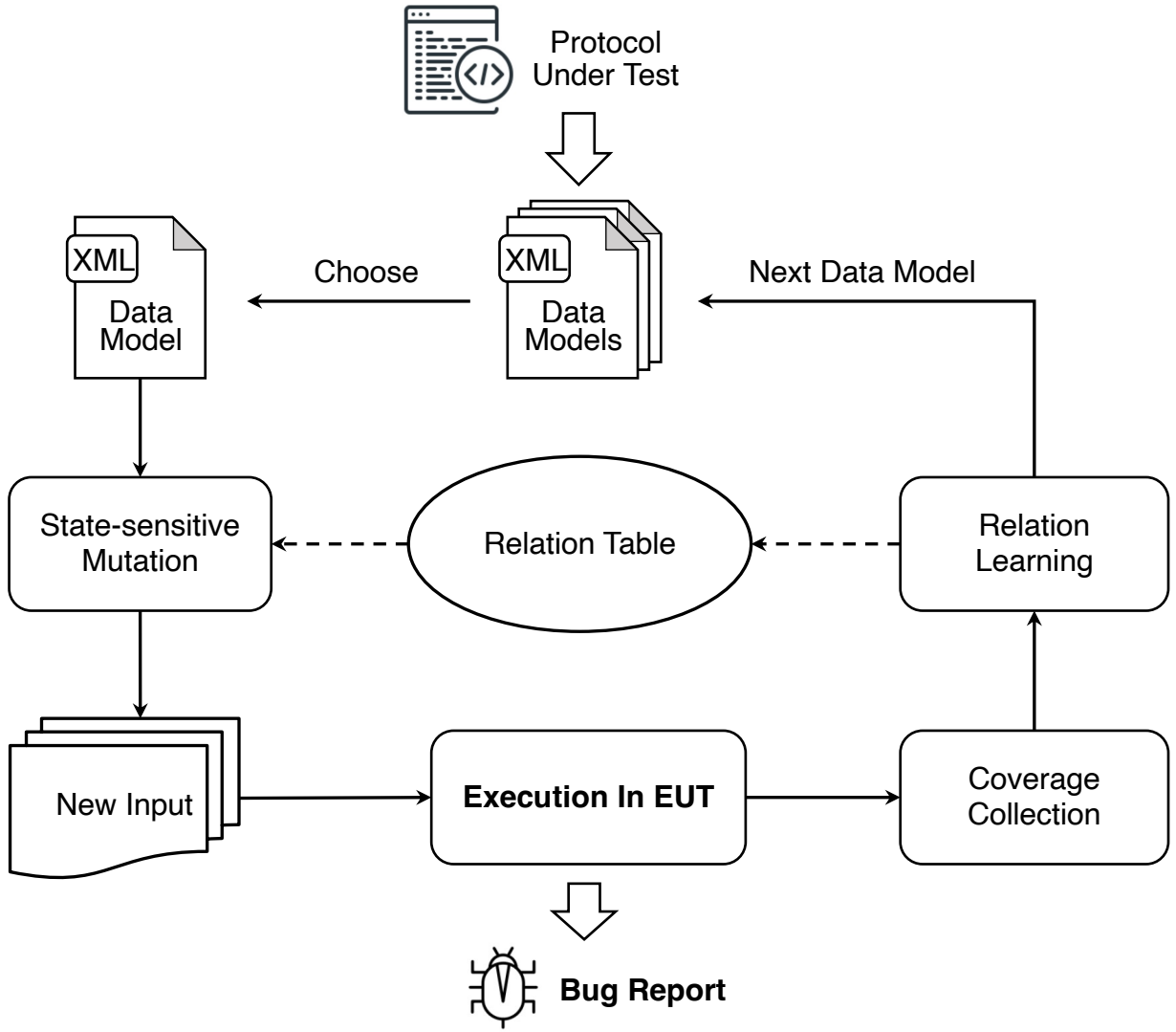
- PAVFuzz starts with user-provided data models of the under-test protocol
- For each model, instead of random mutation, it performs state-sensitive mutation over data elements according to the global relation table.

Overview of PAVFuzz



- PAVFuzz starts with user-provided data models of the under-test protocol
- For each model, instead of random mutation, it performs state-sensitive mutation over data elements according to the global relation table.
- Each generated new input is injected to the endpoint and PAVFuzz monitors the execution to find potential bugs.

Overview of PAVFuzz



- PAVFuzz starts with user-provided data models of the under-test protocol
- For each model, instead of random mutation, it performs state-sensitive mutation over data elements according to the global relation table.
- Each generated new input is injected to the endpoint and PAVFuzz monitors the execution to find potential bugs.
- PAVFuzz collects coverage information for each input and updates the relation table in the relation learning part.

Relation Table

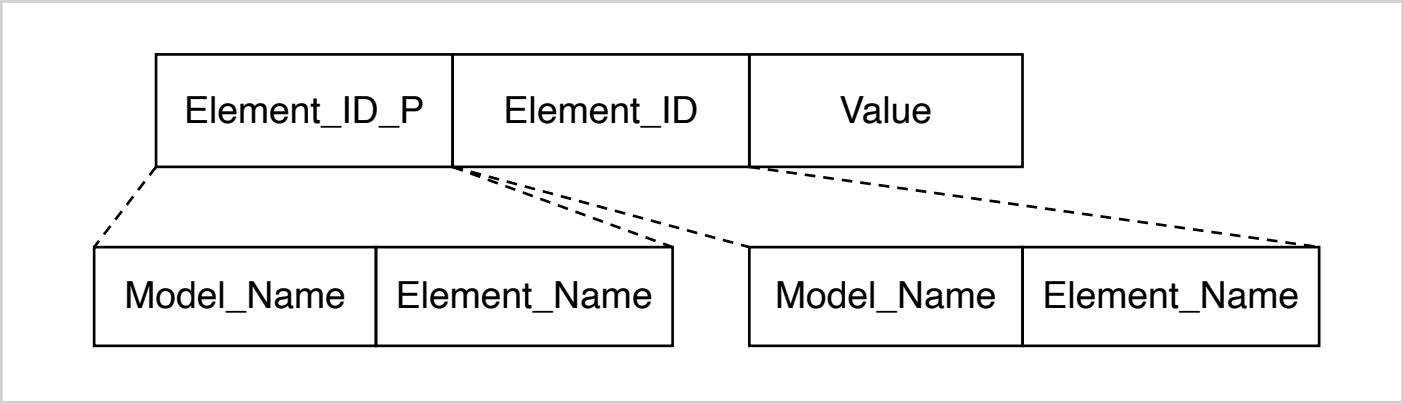


Fig. Structure of each basic cell in the Relation Table.

Relation Table

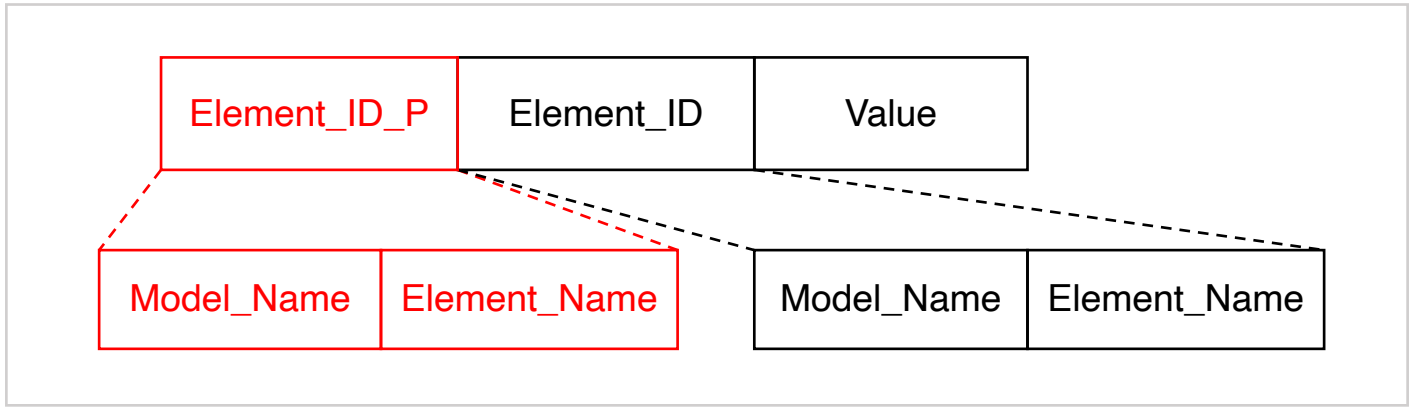


Fig. Structure of each basic cell in the Relation Table.

Example.

- **Element_ID_P**: the ID of data element in the **previous state**
- An element id consists of model name and element name to guarantee the uniqueness

Relation Table

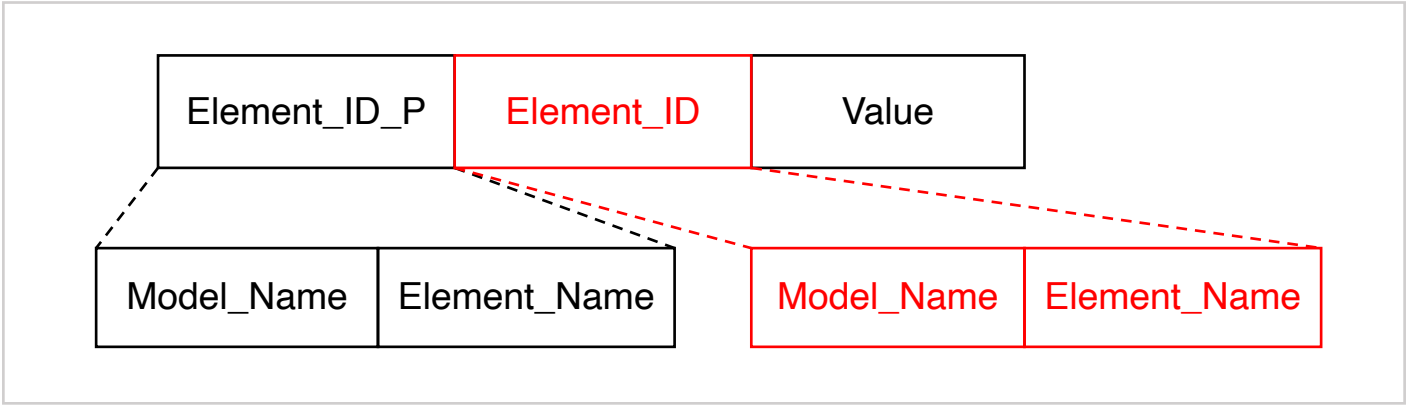


Fig. Structure of each basic cell in the Relation Table.

Example.

- **Element_ID**: the ID of data element in the **successive state**
- State **SPDP --> State SEDP --> State PUB**

Relation Table

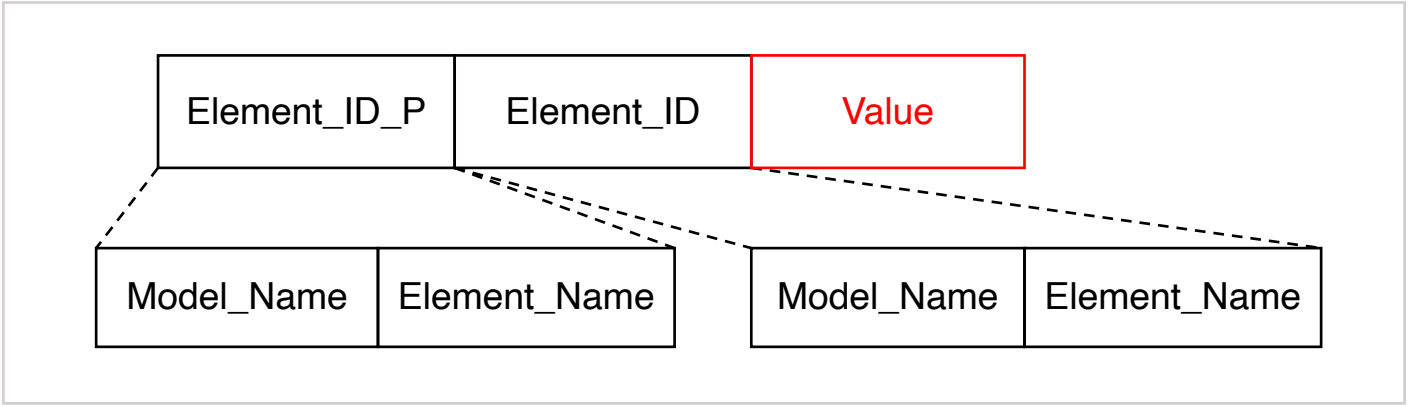
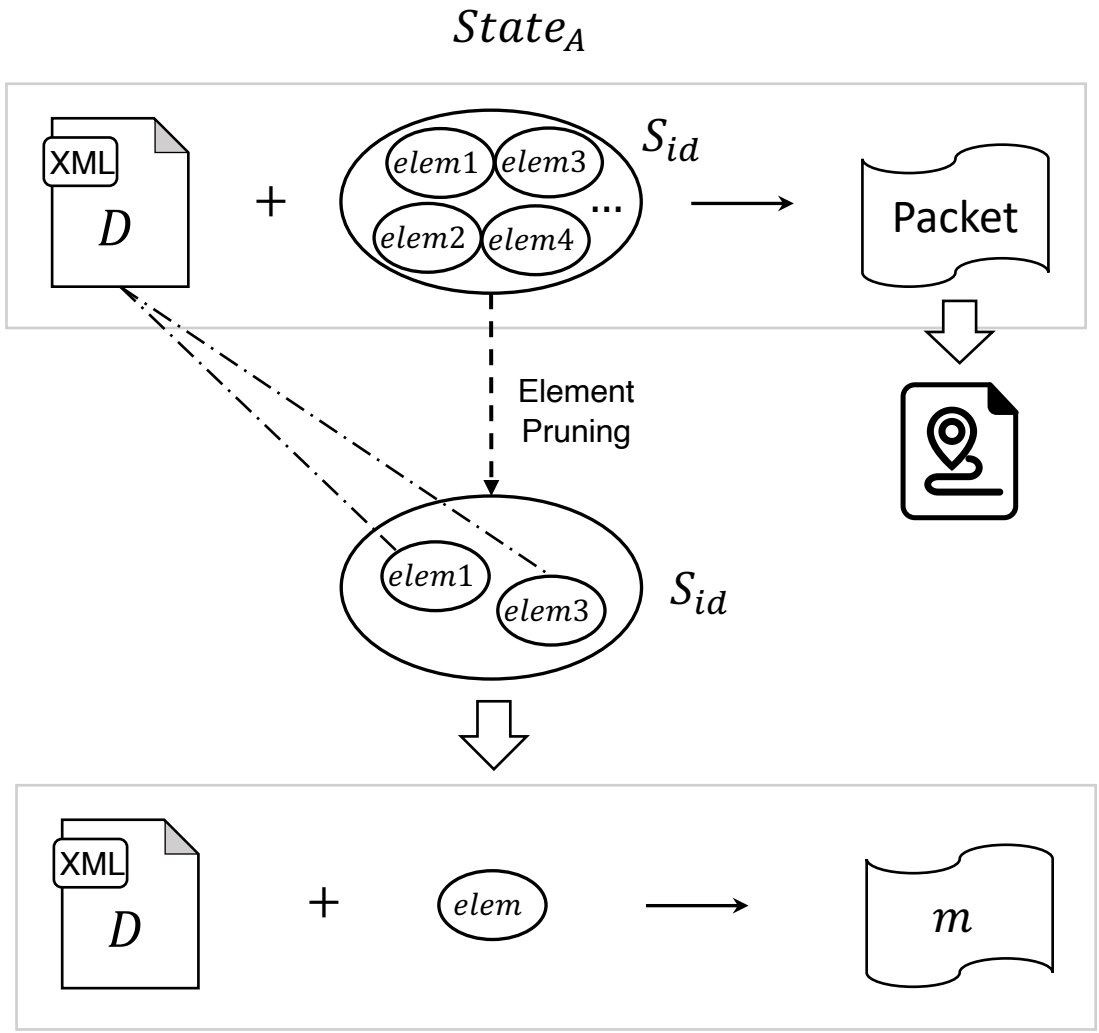


Fig. Structure of each basic cell in the Relation Table.

Example.

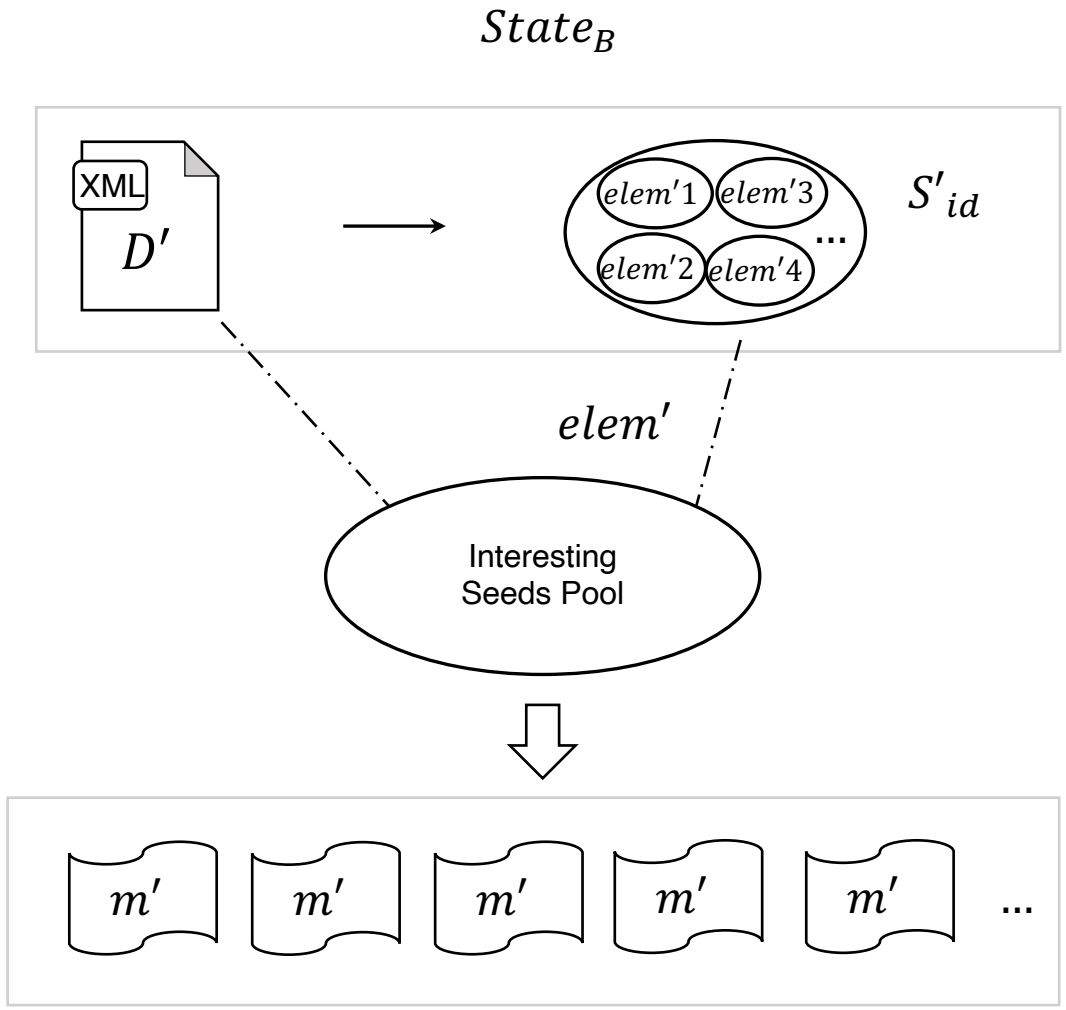
- **Value**: temp value to measure the relation between these two data elements
- Dynamically maintained and updated by PAVFuzz

Relation Learning: To Construct the Relation Table



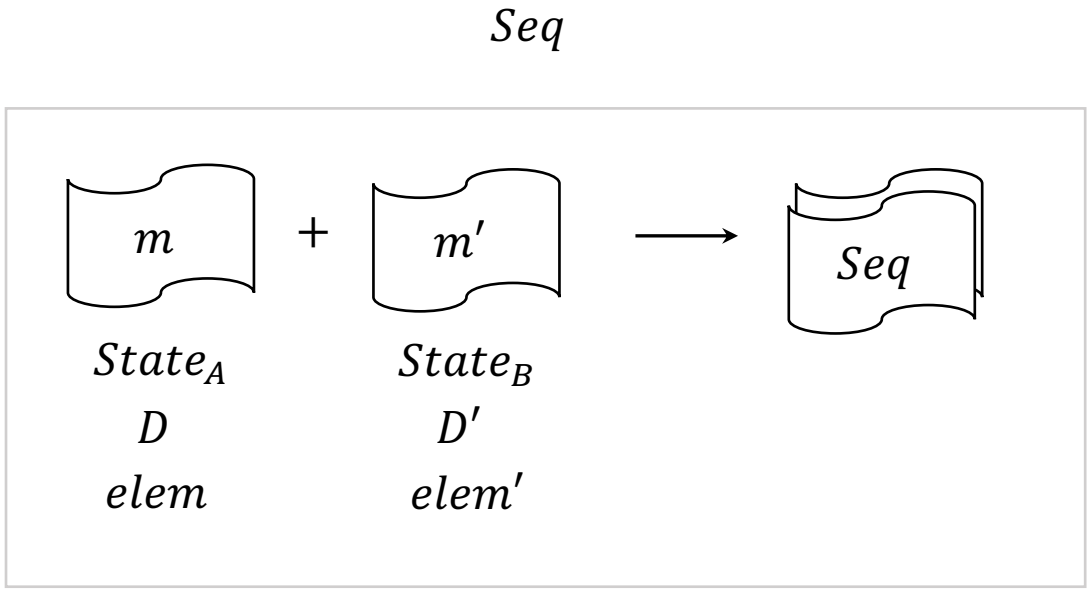
- Triggered when a packet with mutated elements touches new code coverage
- Element pruning to get the minimum set S_{id}
- Generate packet m for each $elem$ in S_{id}

Relation Learning: To Construct the Relation Table



- Triggered when a packet with mutated elements touches new code coverage
- Element pruning to get the minimum set S_{id}
- Generate packet m for each $elem$ in S_{id}
- Refer to the successive state B to pick the set of m' for each $elem'$ from the seeds pool

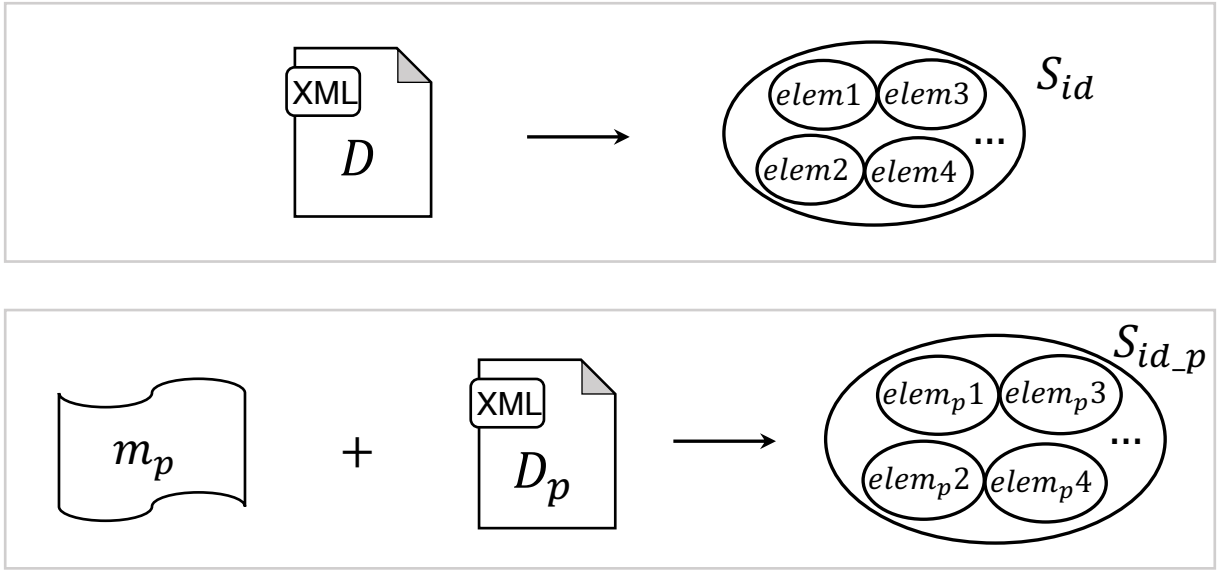
Relation Learning: To Construct the Relation Table



- Triggered when a packet with mutated elements touches new code coverage
- Element pruning to get the minimum set S_{id}
- Generate packet m for each $elem$ in S_{id}
- Refer to the successive state B to pick the set of m' for each $elem'$ from the seeds pool
- Combine each pair of m and m' into new packet sequences $seq < m, m' >$

$$T_r < elem, elem' > increases\ when\ a\ seq < m, m' > coveres\ new\ branches$$

State-Sensitive Mutation: Leveraging the Relations



- Instead of **randomly choosing** elements to mutate, it calculates the different mutation weight W_{elem} for each data element in current state to smartly recognize **the key ones**.

$$\forall elem \in S_{id},\ \ W_{elem} = \sum_{elem_p} T_r < elem_p, elem >,\ elem_p \in S_{id\_p}$$

Experiment Setup

Research Questions

- **R1:** Is PAVFuzz **more efficient in fuzzing** protocols used in autonomous vehicles than start-of-the-art fuzzers?
- **R2:** Can PAVFuzz effectively **expose previously unknown vulnerabilities** in those widely used protocols in autonomous vehicles?

Experiment Setup

Selected Protocols Used in Autonomous Vehicles

- **RTPS – Real-time Publish Subscribe Protocol**

- **SOME/IP – Scalable service-Oriented Middleware over IP**

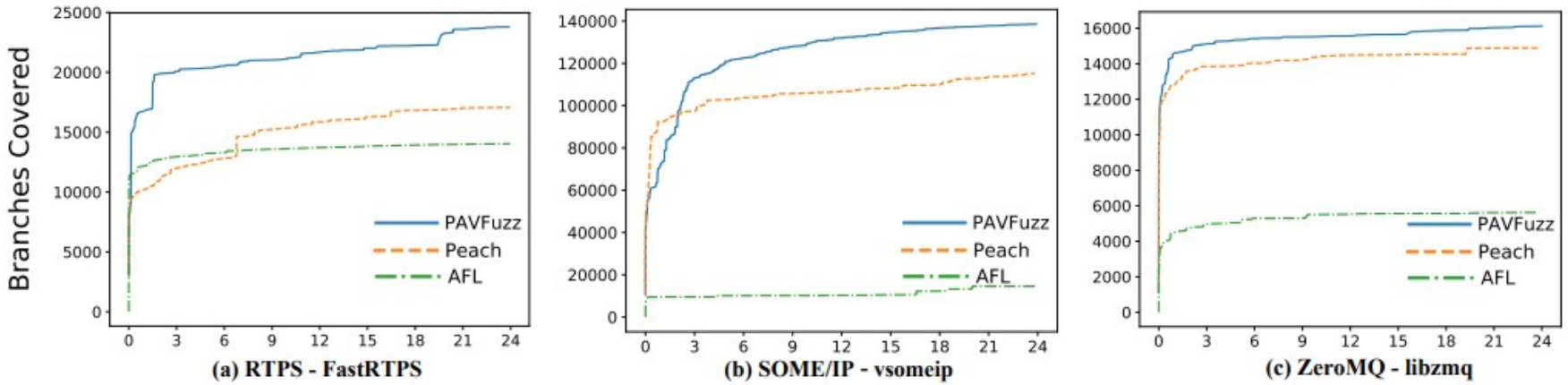
- **ZeroMQ**

Experiment Setup

Selected Protocols Used in Autonomous Vehicles

Protocol	Description
RTPS	RTPS is the standard wire protocol used in Data Distribution Service (DDS), which is adopted in many automatic driving systems such as Adaptive AutoSar, Baidu Apollo, etc .
SOME/IP	SOME/IP is a famous application protocol used in Ethernet-based in-vehicle network systems for service discovery and communication control over ECUs, carmers, radars, and so on.
ZeroMQ	ZeroMQ is a lightweight protocol for distributed communication. It has been adopted as alternative protocol in ROS2 (Robot Operating System), a prototype system for automotive driving.

Efficiency of Fuzzing



Subject	AFL	Peach	PAVFuzz	$\mathcal{I}_{\mathcal{A}}$	$\mathcal{I}_{\mathcal{P}}$
FastRTPS	14034	17107	23784	+69.47%	+39.03%
vsomeip	14562	115147	138548	+851.44%	+20.32%
libzmq	5621	14894	16114	+186.67%	+8.19%
AVERAGE				+369.19%	+22.51%

AFL +369.19% ↑ Peach +22.51% ↑

Previous Unknown Vulnerabilities

Subject	Vulnerability	AFL	Peach	PAVFuzz
FastRTPS	stack-buffer-overflow-1	✗	✓	✓
	stack-buffer-overflow-2	✗	✓	✓
	stack-buffer-overflow-3	✗	✗	✓
	stack-buffer-overflow-4	✗	✓	✓
	stack-buffer-overflow-5	✗	✓	✓
	stack-buffer-overflow-6	✗	✗	✓
	heap-buffer-overflow-1	✗	✗	✓
	heap-buffer-overflow-2	✗	✗	✓
	heap-buffer-overflow-3	✗	✗	✓
vsomeip	allocate-out-of-memory	✗	✓	✓
	heap-buffer-overflow	✗	✗	✓
libzmq	allocate-memory-failure	✓	✓	✓
Total	12	1/12	6/12	12/12

DEC 5 - 9, 2021 ◆ San Francisco, California

Thanks for your attention!