LinuxTesting.org

Part I. Hunting for Bugs

Vadim Mutilin

ISP RAS

Institute for System Programming of the Russian Academy of Sciences

About Us
- About Center
- Our Team
- News
- Partners
- Contacts

Projects
- Linux Kernel Space Verification
- LSB Infrastructure
- Testing Technologies
- Tests and Frameworks
- Portability Tools

Results
- Contribution
  - Problems in Linux Kernel
  - Problems in Libraries
  - Problems in Standards
- Publications
- Events

Vadim Mutilin
- My account
- User list
- Create content
- Feed aggregator
- Log out

Problems in Linux Kernel

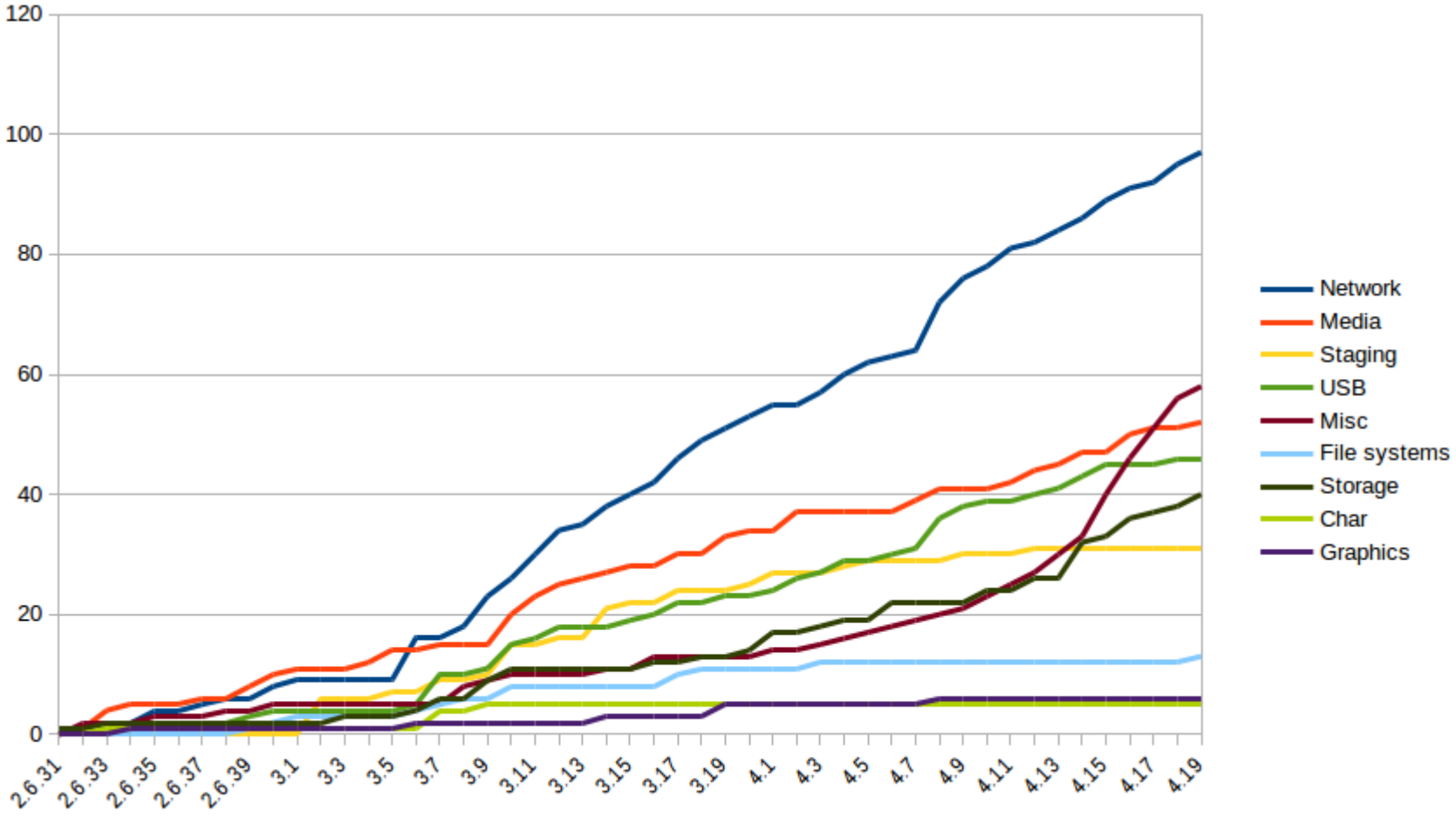
View | Edit | Translate

This section contains information about problems in Linux kernel found within Linux Driver Verification program as well as within KEDR and Linux File System Verification projects.

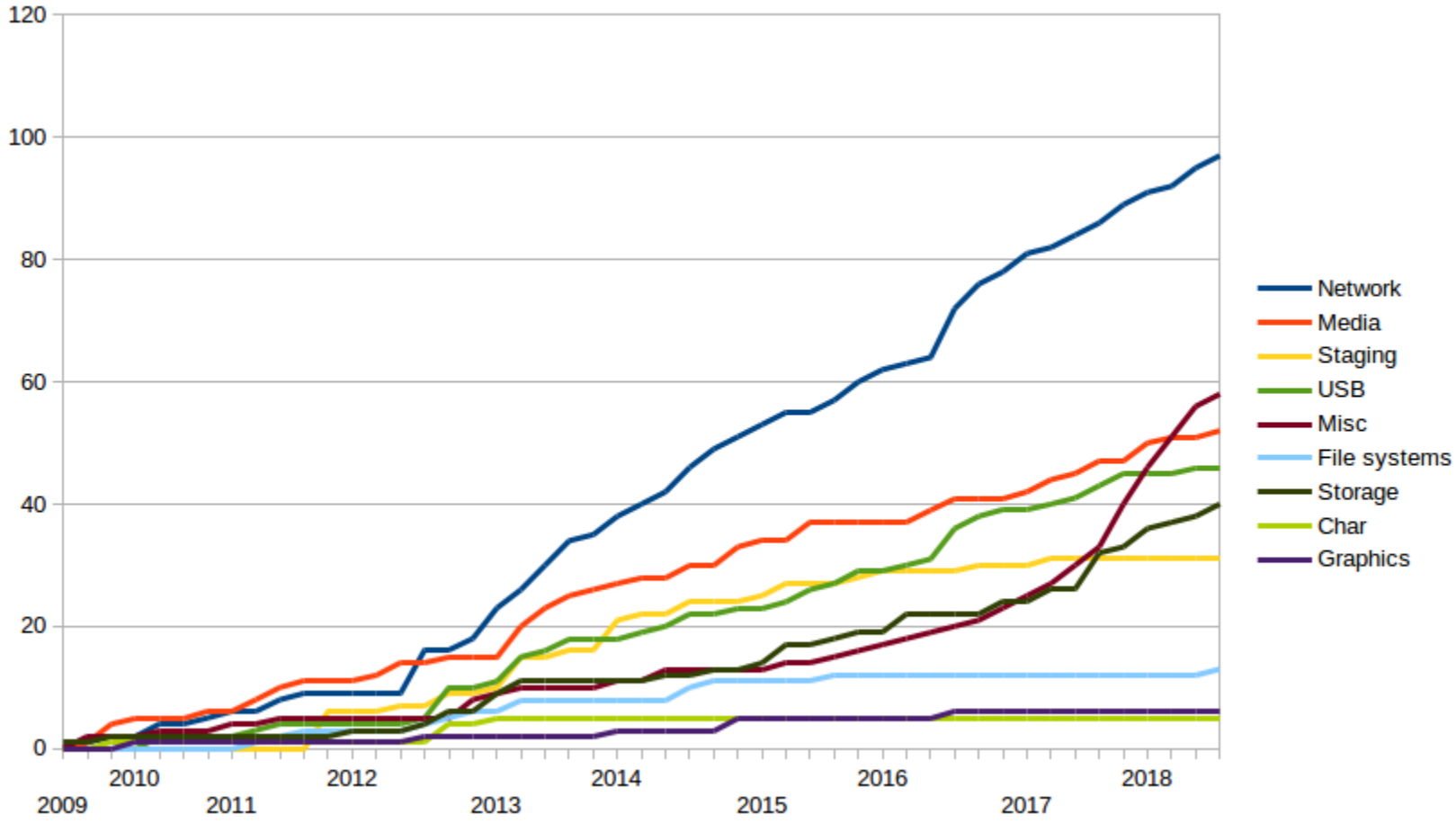
Click on a problem number for detailed description. Click on a column header to change the sorting order.

No.	Type	Brief	Added on	Accepted	Status
L0351	Crash	regulator: tps65217: NULL pointer dereference on probe	2019-09-26	https://lkml.org/lkml/2018/7/27/661 commit	Fixed in kernel v4.19-rc1
L0350	Crash	scsi: 3ware: fix return 0 on the error path of probe	2019-09-25	https://lkml.org/lkml/2018/7/27/655 commit	Fixed in kernel v4.19-rc1
L0349	Crash	gpio: ml-ioh: buffer underwrite on probe error path	2019-09-24	https://lkml.org/lkml/2018/7/23/949 commit	Fixed in kernel v4.19-rc1
L0348	Crash	firmware: vpd: incorrect section enabled flag on vpd_section_destroy	2019-09-20	https://lkml.org/lkml/2018/7/23/944 commit	Fixed in kernel v4.19-rc1
L0347	Leak	mtd: spi-nor: nxp-spifi: missing release of flash_np in nxp_spifi_probe	2019-09-20	https://lkml.org/lkml/2018/5/9/579 commit	Fixed in kernel v4.19-rc1
L0346	Leak	media: dw2102: memleak in dw2102_probe	2019-09-20	https://lkml.org/lkml/2018/7/23/964 commit	Fixed in kernel v4.19-rc1
L0345	Leak	uwb: hwa-rc: memory leak at hwarc_probe	2019-09-20	https://lkml.org/lkml/2018/7/6/412 commit	Fixed in kernel v4.19-rc1
L0344	Crash	tty: rocket: possible buffer overwrite on register_PCI	2019-09-20	https://lkml.org/lkml/2018/7/27/644 commit	Fixed in kernel v4.19-rc1
L0343	Leak	net: mdio-mux: bcm-iproc: wrong getter and setter pair	2019-09-20	https://lkml.org/lkml/2018/7/27/772 commit	Fixed in kernel

Bugs Found for Subsystems



Bugs Found for Subsystems



About Us
- About Center
- Our Team
- News
- Partners
- Contacts

Projects
- Linux Kernel Space Verification
- LSB Infrastructure
- Testing Technologies
- Tests and Frameworks
- Portability Tools

Results
- Contribution
- Publications
- Events

Vadim Mutilin
- My account
- User list
- Create content
- Feed aggregator
- Log out

Problems in Linux Kernel Found by CPAchecker

View | Edit | Translate

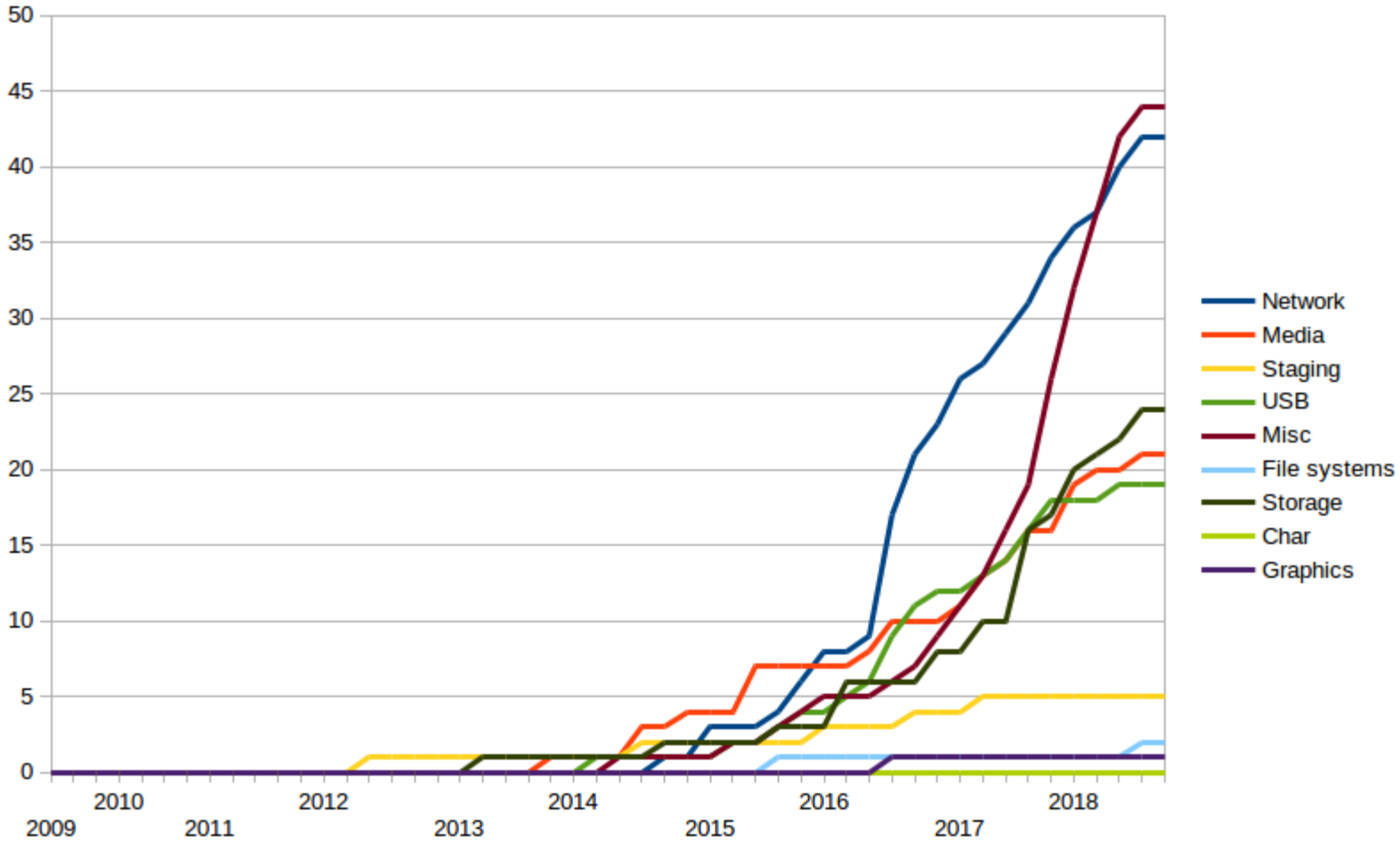
This section contains information about problems in Linux kernel found within Linux Driver Verification project with CPAchecker engine.

Click on a problem number for detailed description. Click on a column header to change the sorting order.

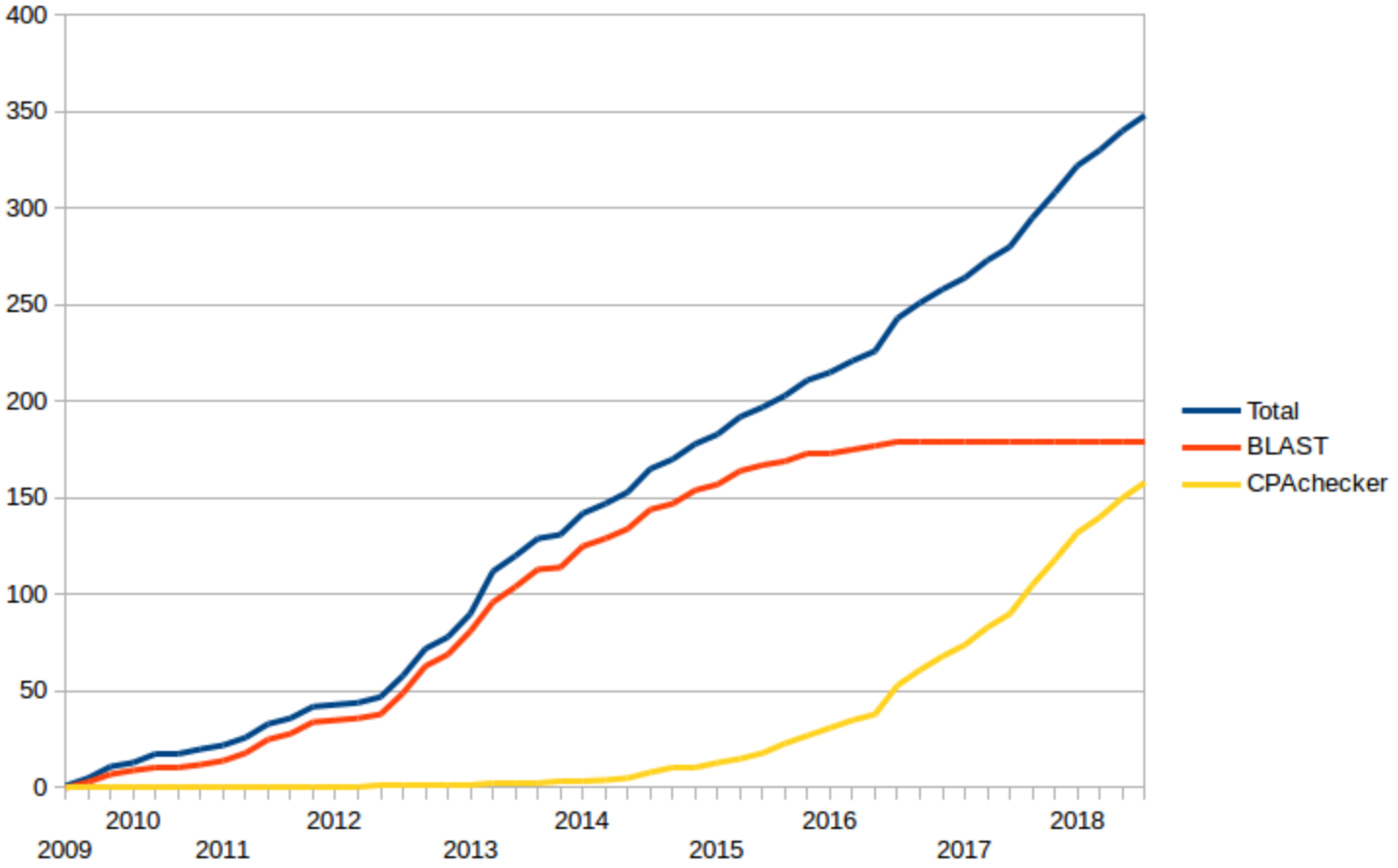
There are **158 bugs** found by **CPAchecker**.

No.	Type	Brief	Added on	Accepted	Status	Trace
L0351	Crash	regulator: tps65217: NULL pointer dereference on probe	2019-09-26	https://lkml.org/lkml/2018/7/27/661 commit	Fixed in kernel v4.19-rc1	...
L0350	Crash	scsi: 3ware: fix return 0 on the error path of probe	2019-09-25	https://lkml.org/lkml/2018/7/27/655 commit	Fixed in kernel v4.19-rc1	...
L0349	Crash	gpio: ml-ioh: buffer underwrite on probe error path	2019-09-24	https://lkml.org/lkml/2018/7/23/949 commit	Fixed in kernel v4.19-rc1	...
L0348	Crash	firmware: vpd: incorrect section enabled flag on vpd_section_destroy	2019-09-20	https://lkml.org/lkml/2018/7/23/944 commit	Fixed in kernel v4.19-rc1	...
L0347	Leak	mtd: spi-nor: nxp-spifi: missing release of flash_np in nxp_spifi_probe	2019-09-20	https://lkml.org/lkml/2018/5/9/579 commit	Fixed in kernel v4.19-rc1	...
L0346	Leak	media: dw2102: memleak in dw2102_probe	2019-09-20	https://lkml.org/lkml/2018/7/23/964 commit	Fixed in kernel v4.19-rc1	...
L0345	Leak	uwb: hwa-rc: memory leak at hwarc_probe	2019-09-20	https://lkml.org/lkml/2018/7/6/412 commit	Fixed in kernel v4.19-rc1	...

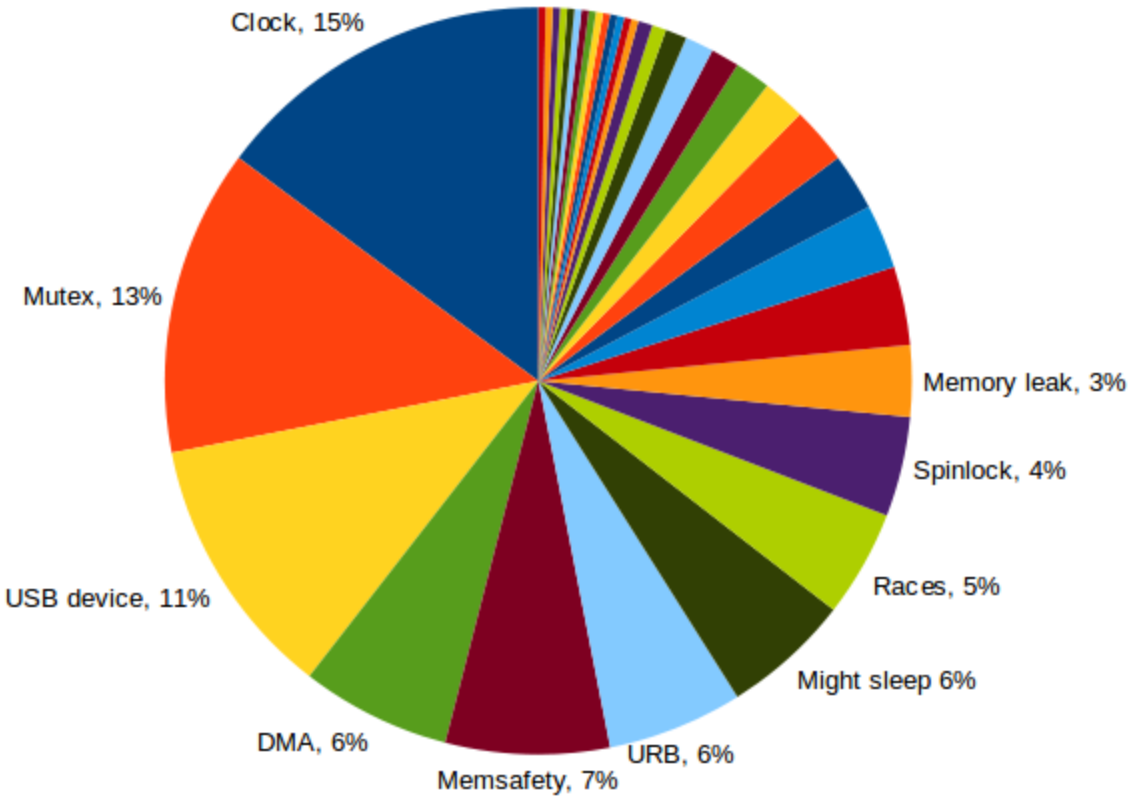
Bugs Found by CPAchecker



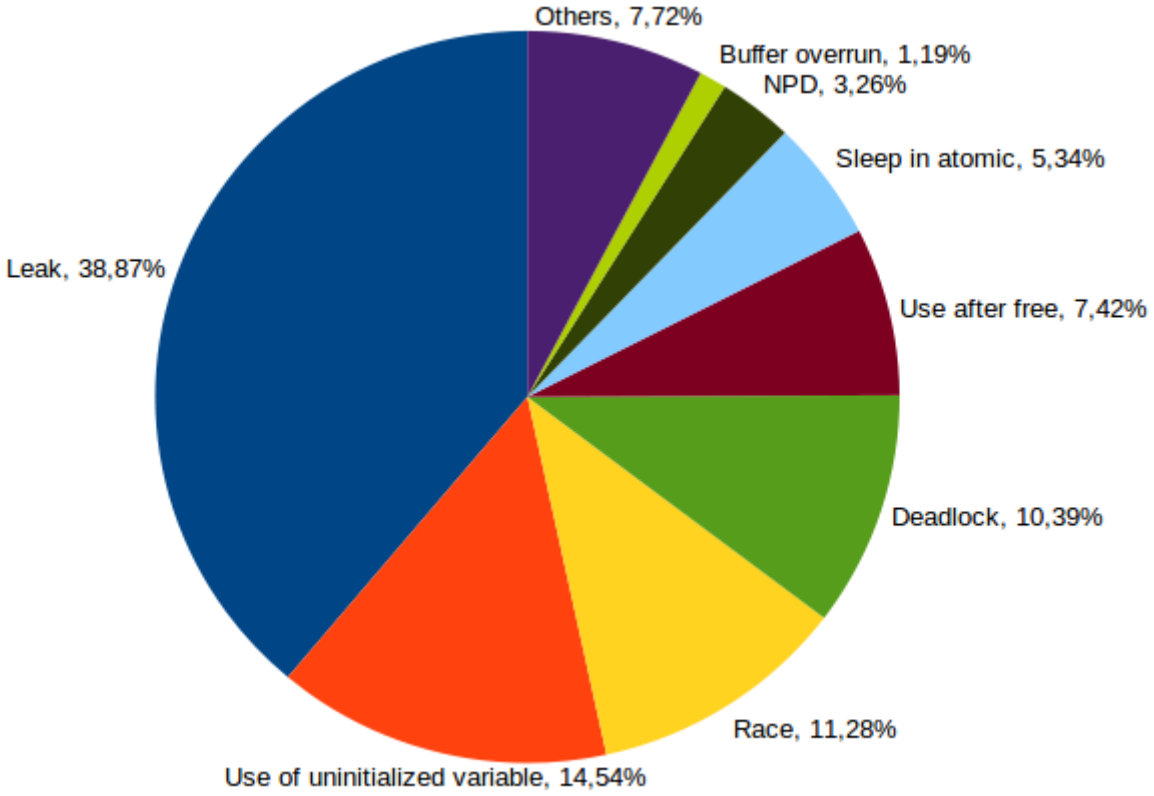
Total Bugs Found



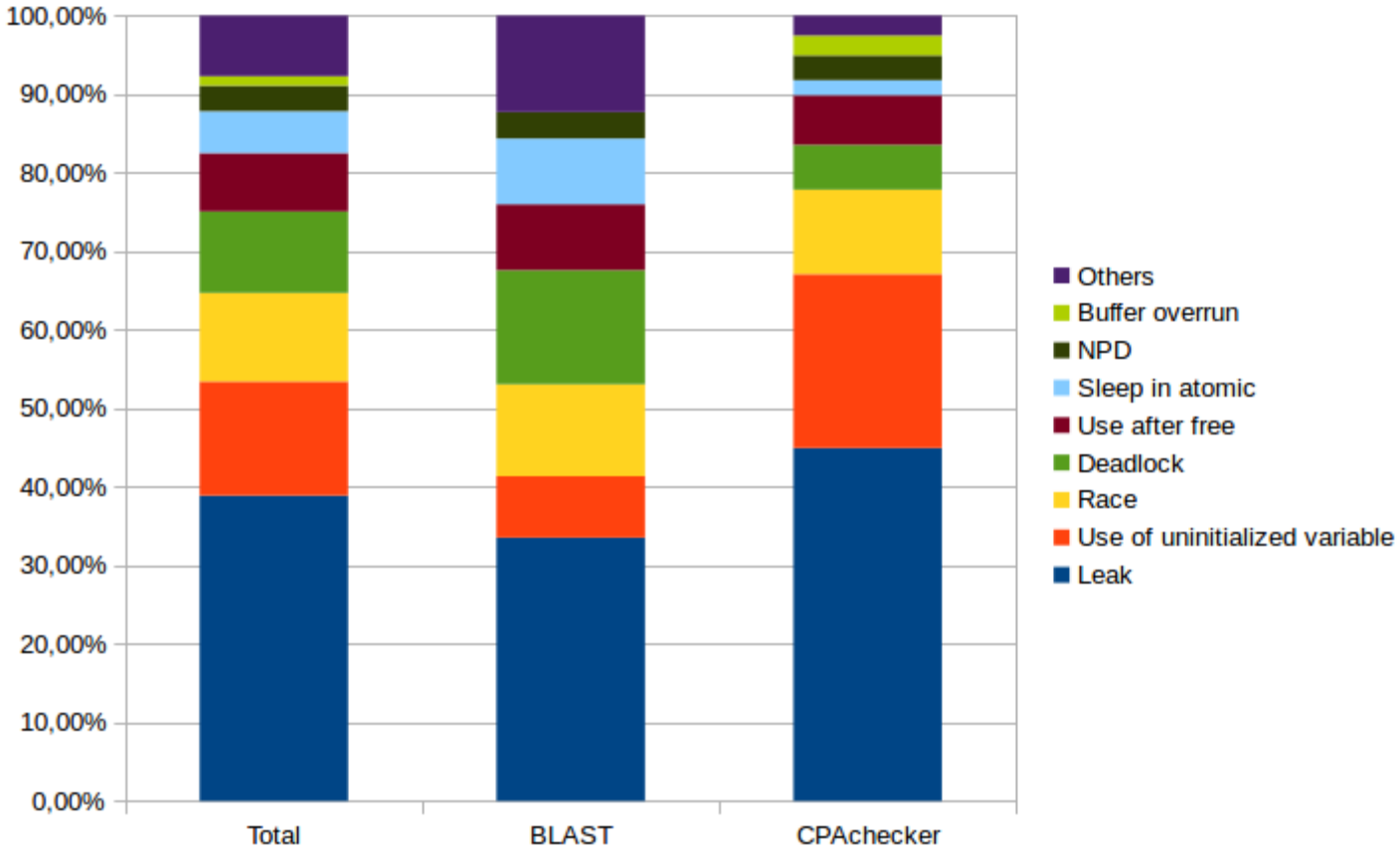
Top 10 of 35 Rules



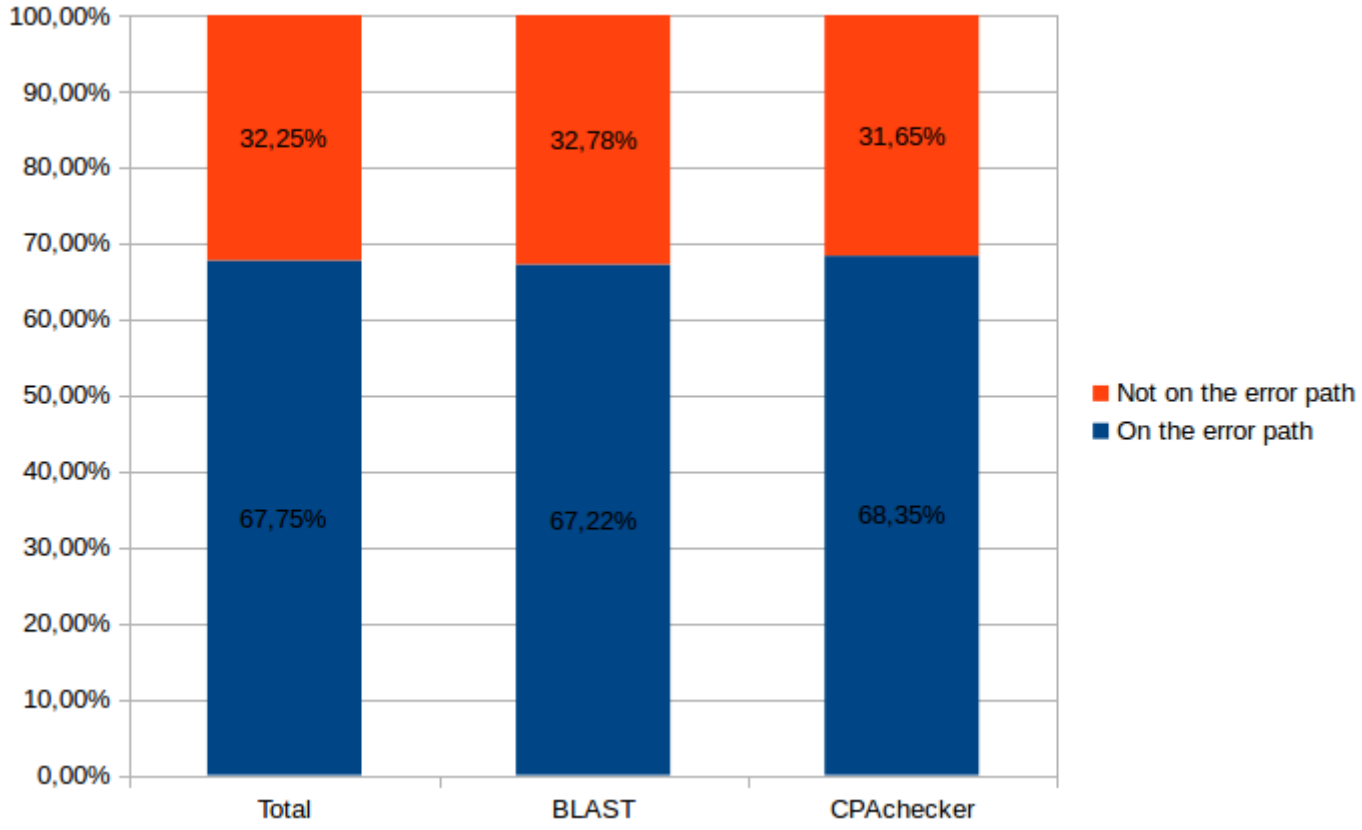
Consequences



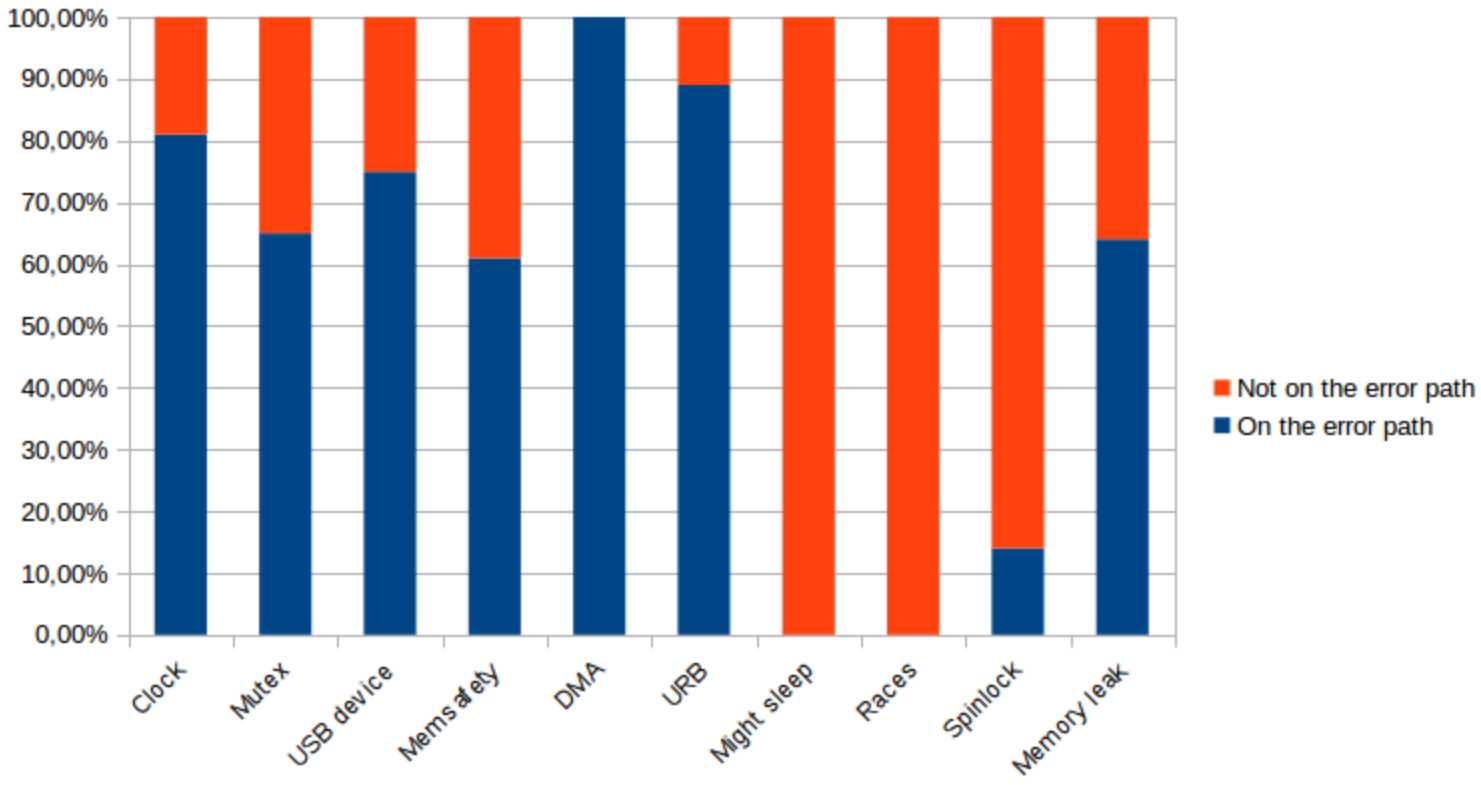
Consequences (by the tool)



On the Error Path? (by the tool)



On the Error Path? (for top 10 rules)



LinuxTesting.org

Part II. ARINC to AADL

Sergey Lesovoy

ISP RAS

Institute for System Programming of the Russian Academy of Sciences

ARINC 653

From Wikipedia, the free encyclopedia

ARINC 653 (Avionics Application Standard Software Interface) is a software specification for space and time partitioning in safety-critical avionics real-time operating systems (RTOS). It allows the hosting of multiple applications of different software levels on the same hardware in the context of an Integrated Modular Avionics architecture.[1]

- Partition & process management

```
process_attrs.ENTRY_POINT = first_process;
strncpy(process_attrs.NAME, "process 1", sizeof(PROCESS_NAME_TYPE));

CREATE_PROCESS(&process_attrs, &pid, &ret);
```

...

```
SET_PARTITION_MODE(NORMAL, &ret);
```

- Inter & intra partition communication

```
// create ports
CREATE_QUEUING_PORT("QP1", 64, 10, DESTINATION, FIFO, &QP1, &ret);
```

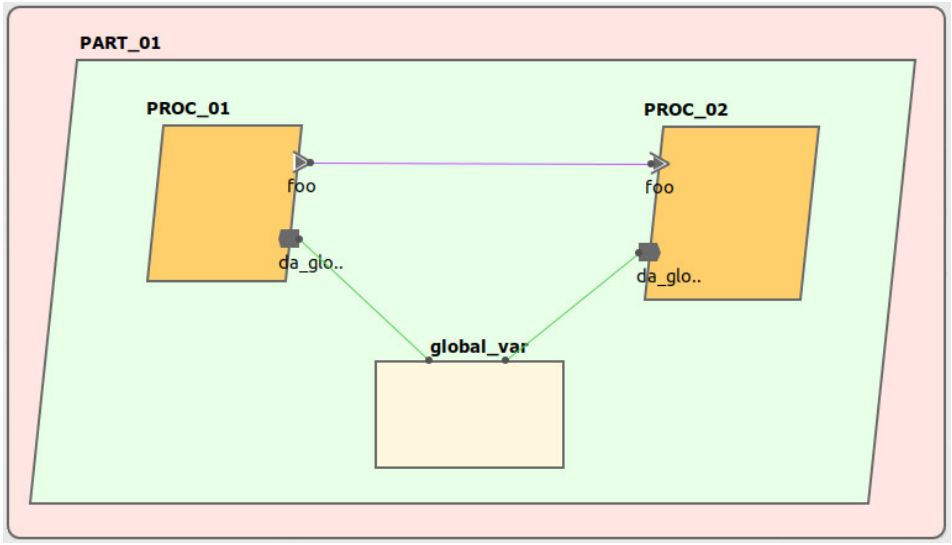
...

```
MESSAGE_SIZE_TYPE len;
RECEIVE_QUEUING_MESSAGE(QP1, INFINITE_TIME_VALUE, (MESSAGE_ADDR_TYPE) &msg, &len, &ret);
```

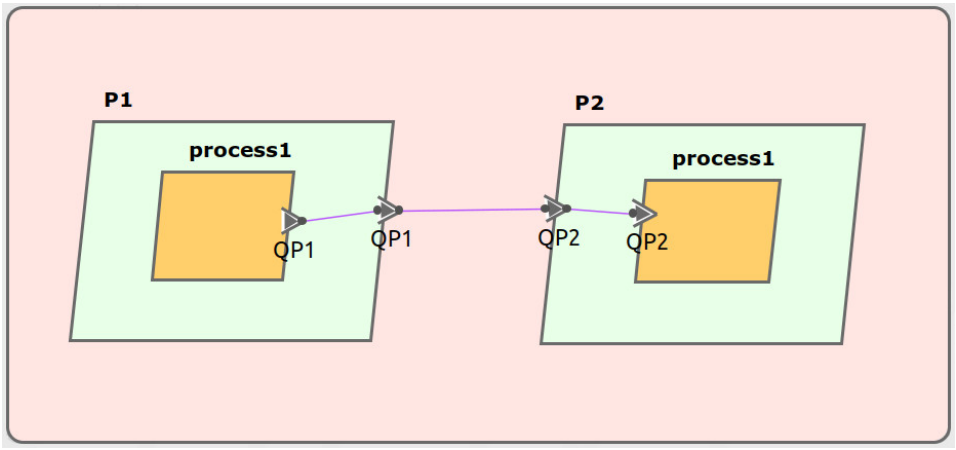
- etc

Architecture Analysis and Design Language (AADL)

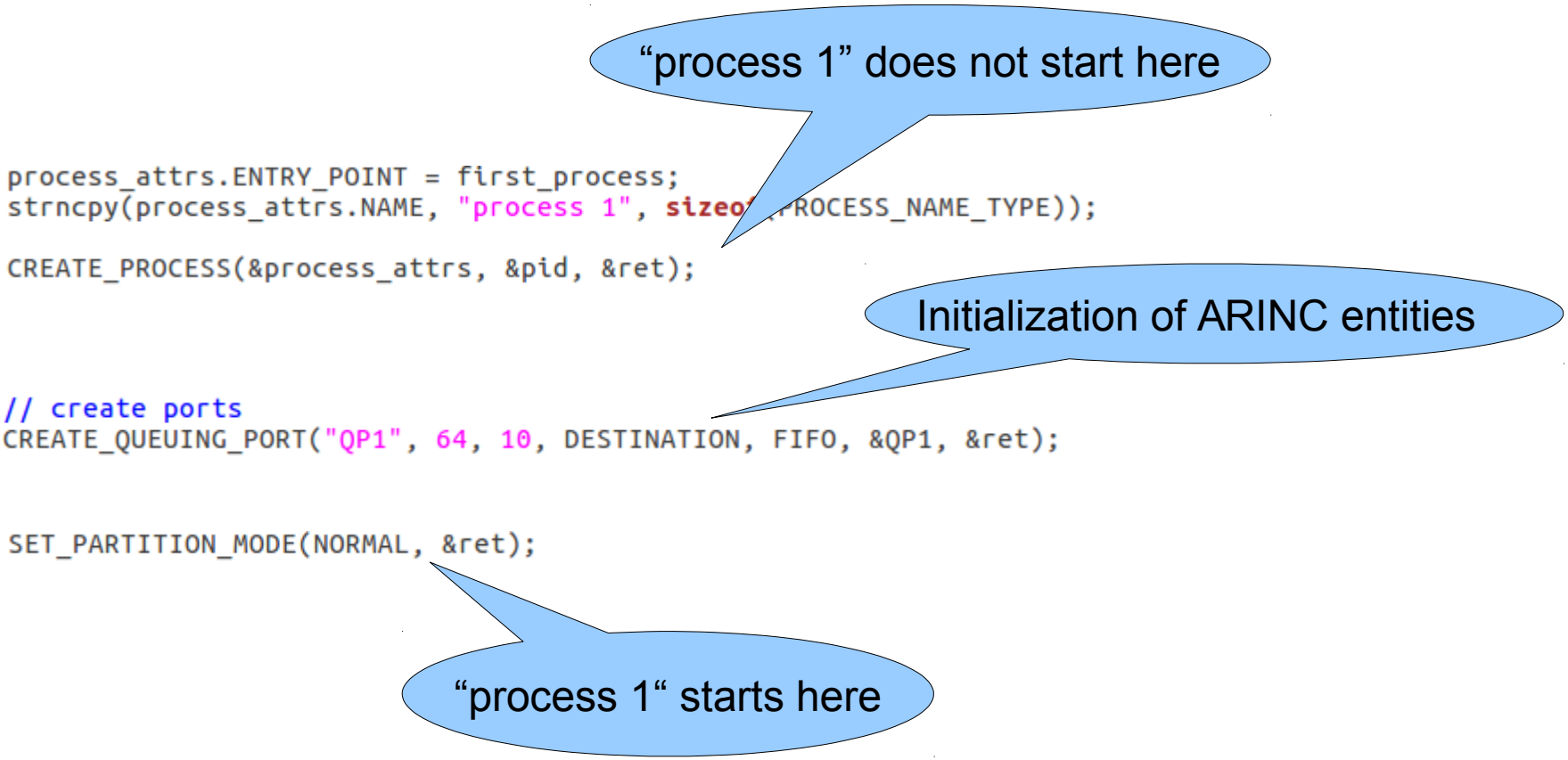
Example 1.
Communicating with intra partition port and global variable



Example 2.
Communicating with inter partition port



ARINC processes



ARINC processes Solution 1

- Preliminary Value analysis collecting set of process function pointers
- Inserting function calls explicitly

```
process_attrs.ENTRY_POINT = first_process;
strncpy(process_attrs.NAME, "process 1", sizeof(PROCESS_NAME_TYPE));

CREATE_PROCESS(&process_attrs, &pid, &ret);

// create ports
CREATE_QUEUING_PORT("QP1", 64, 10, DESTINATION, FIFO, &QP1, &ret);

SET_PARTITION_MODE(NORMAL, &ret);

first_process();
```

Start “process 1”

ARINC processes Solution 2

- A model with nondeterministic choice

```
SYSTEM_ENTRY_TYPE nondet_entry_point = 0;

void CREATE_PROCESS (
    PROCESS_ATTRIBUTE_TYPE  *attributes,
    PROCESS_ID_TYPE         *process_id,
    RETURN_CODE_TYPE        *return_code)
{
    if(__VERIFIER_nondet_int()) {
        nondet_entry_point = attributes->ENTRY_POINT;
    }
}

void SET_PARTITION_MODE (...)
{
    (*nondet_entry_point)();
}
```

Save nondeterministically

Call saved pointer

ARINC entities

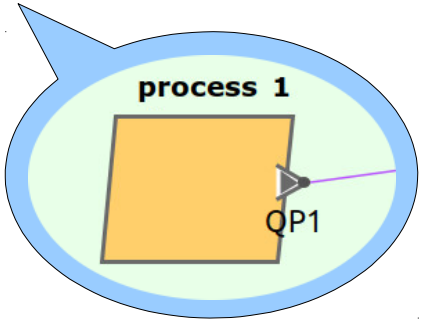
```
process_attrs.ENTRY_POINT = first_process;
strncpy(process_attrs.NAME, "process 1", sizeof

CREATE_PROCESS(&process_attrs, &pid, &ret);

// create ports
CREATE_QUEUING_PORT("QP1", 64, 10, DESTINATION, FIFO, &QP1, &ret);
SET_PARTITION_MODE(NORMAL, &ret);
```

Creation of process with name “process 1” and function pointer first_process. Identifier is stored in variable pid

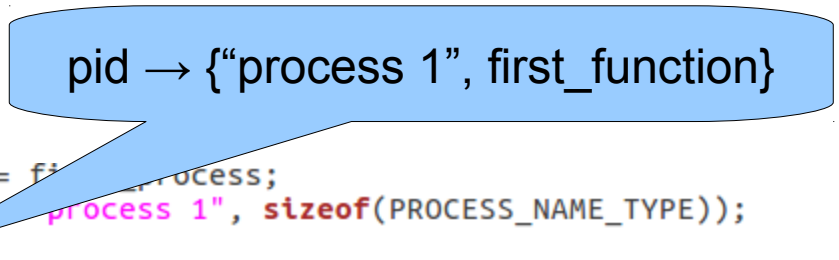
Creation of port entity with name “QP1” Identifier is stored in variable QP1

```
void first_process() {

...
    MESSAGE_SIZE_TYPE len;
    RECEIVE_QUEUING_MESSAGE(QP1, INFINITE_TIME_VALUE, (MESSAGE_ADDR_TYPE) &msg, &len, &ret);
```

Entering “process 1”



ARINC entities Solution




```
process_attrs.ENTRY_POINT = fi      ocess;
strncpy(process_attrs.NAME      process 1", sizeof(PROCESS_NAME_TYPE));

CREATE_PROCESS(&process_attrs, &pid, &ret);

// create ports
CREATE_QUEUING_PORT("QP1", 64, 10, DESTINATION, FIFO, &QP1, &ret);
SET_PARTITION_MODE(NORMAL, &ret);
```

```
void first_process() {

...
    MESSAGE_SIZE_TYPE len;
    RECEIVE_QUEUING_MESSAGE(QP1, INFINITE_TIME_VALUE, (MESSAGE_ADDR_TYPE) &msg, &len, &ret);
```

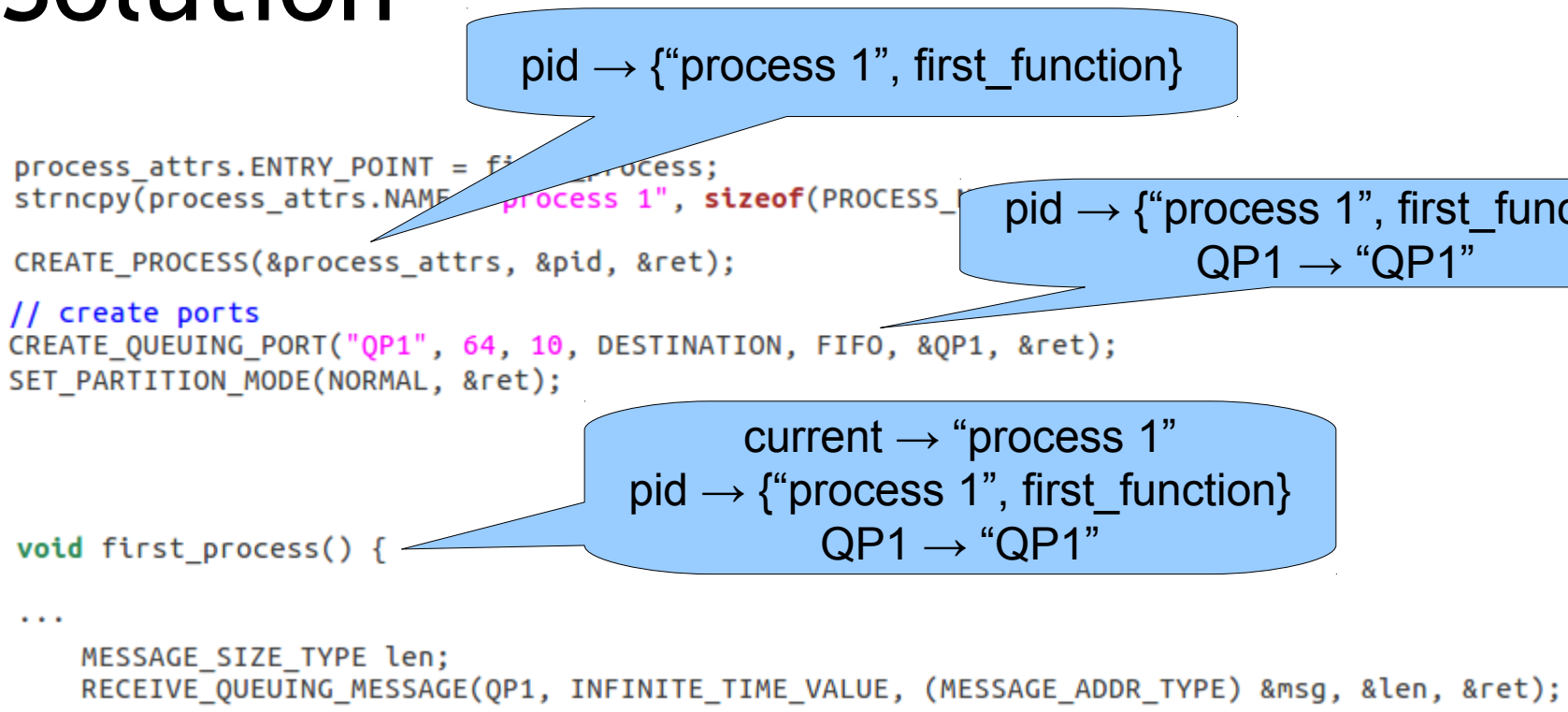
ARINC entities Solution

pid → {“process 1”, first_function}

pid → {“process 1”, first_function}
QP1 → “QP1”

```
process_attrs.ENTRY_POINT = fi      ocess;
strncpy(process_attrs.NAME      process 1", sizeof(PROCESS_

CREATE_PROCESS(&process_attrs, &pid, &ret);

// create ports
CREATE_QUEUING_PORT("QP1", 64, 10, DESTINATION, FIFO, &QP1, &ret);
SET_PARTITION_MODE(NORMAL, &ret);



void first_process() {

...
    MESSAGE_SIZE_TYPE len;
    RECEIVE_QUEUING_MESSAGE(QP1, INFINITE_TIME_VALUE, (MESSAGE_ADDR_TYPE) &msg, &len, &ret);
```

ARINC entities Solution



ARINC entities Solution

pid → {“process 1”, first_function}

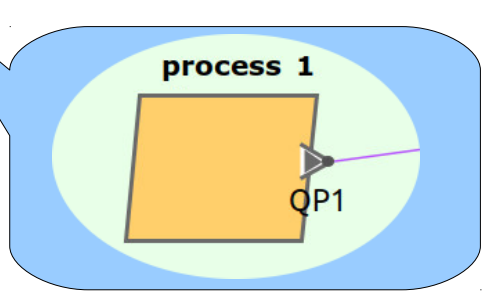
```
process_attrs.ENTRY_POINT = fi      ocess;
strncpy(process_attrs.NAME      process 1", sizeof(PROCESS_

CREATE_PROCESS(&process_attrs, &pid, &ret);
```

pid → {“process 1”, first_function}
QP1 → “QP1”

```
// create ports
CREATE_QUEUING_PORT("QP1", 64, 10, DESTINATION, FIFO, &QP1, &ret);
SET_PARTITION_MODE(NORMAL, &ret);
```

current → “process 1”
pid → {“process 1”, first_function}
QP1 → “QP1”

```
void first_process() {

...
    MESSAGE_SIZE_TYPE len;
    RECEIVE_QUEUING_MESSAGE(QP1, INFINITE_TIME_VALUE, (MESSAGE_ADDR_TYPE) &msg, &len, &ret);
```



Collect values only on reachable paths

```
process_attrs.ENTRY_POINT = first_process;
strncpy(process_attrs.NAME, "process 1", sizeof(PROCESS_NAME_TYPE));

CREATE_PROCESS(&process_attrs, &pid, &ret);

// create ports
CREATE_QUEUING_PORT("QP1", 64, 10, DESTINATION, FIFO, &QP1, &ret);
SET_PARTITION_MODE(NORMAL, &ret);
```

```
void first_process() {

...
    MESSAGE_SIZE_TYPE len;
    RECEIVE_QUEUING_MESSAGE(QP1, INFINITE_TIME_VALUE, (MESSAGE_ADDR_TYPE) &msg, &len, &ret);
```

1. The path should be reachable
2. Get the value

Collect values only on reachable paths Solution. Refinement

```
process_attrs.ENTRY_POINT = first_process;
strncpy(process_attrs.NAME, "process 1", sizeof(PROCESS_NAME_TYPE));

CREATE_PROCESS(&process_attrs, &pid, &ret);

// create ports
CREATE_QUEUING_PORT("QP1", 64, 10, DESTINATION, FIFO, &QP1, &ret);
SET_PARTITION_MODE(NORMAL, &ret);
```

```
void first_process() {

...
    MESSAGE_SIZE_TYPE len;
    RECEIVE_QUEUING_MESSAGE(QP1, INFINITE_TIME_VALUE, (MESSAGE_ADDR_TYPE) &msg, &len, &ret);
```

Mark as target state (violation)
Take the value from Value analysis

CPAchecker ARINC2AADL

- ARINC processes
  - Solution 1. Preliminary analysis – requires modification of CFA
  - Solution 2. Nondeterministic choice – sound for sequential analysis only
- ARINC entities
  - Solution. Extension of Value analysis – supports pointers only heuristically
  - Generalize for other analyses?
- Collect values on reachable paths
  - Value Analysis – supports pointers only heuristically
  - Predicate Analysis – how to exclude undefined values?

LinuxTesting.org

Part III. Topics

ISP RAS

Institute for System Programming of the Russian Academy of Sciences

Topics (unsorted)

- Collect data values with predicate analysis
- Correctness witness visualization
- Stepwise input program simplification and debugging of CPAchecker
- Type and BnB regions for array encoding in predicate analysis
- CPALockator
  - Support for atomic access primitives
  - Support for interrupts model
  - Shared analysis with refinement
  - Support for message passing
  - Support for control dependencies
- Checking memsafety properties for multithreaded programs
- Simplifying input source code for the verification (CIL-less)
- Loop iterations abstraction and refinement
- Generation of exploits
- Checking for undefined behavior with symbolic memory graphs
- Local path refinement selection in BAM
- On-demand memory for predicate analysis
- Runtime learning of environment models

LinuxTesting.org

Thank you!

Vadim Mutilin
http://linuxtesting.org/project/ldv

ISP RAS
Institute for System Programming of the Russian Academy of Sciences