Type – II Embedded systems utilize a RTOS, which can have a portable layer interacting with the HW peripherals.

- Non-essential peripherals (like GPIO) can send malicious data from the external world.
- Essential peripherals (like CLx) can send malformed data to the SW stack.

**EXISTING REHOSTING APPROACHES**

Many existing approaches propose rehosting solutions, but they require one or more of the below:
- Real hardware
- Peripheral modeling
- Base emulator support

Our approach does not require these!

**AUTOMATED FRAMEWORK**

**MMIO in Code:**

```c
// in code

MMIO: Read from MMIO
MMIO: Write to MMIO
MMIO: Read from MMIO
MMIO: Write to MMIO
```

**REAL-TIME FUZZING**

AFL++ Persistent Mode:

- Allows continuous fuzzing of a target with new inputs without having to reftork the target every time.

**AUTOMATED PERIPHERAL HANDLING**

Automated phase:

- Sources and compile time flag information is gathered
- Portable layer of RTOS is changed to Linux one
- Assembly snippets are stripped.

Semi-automated phase:

- User iteratively fixes errors to get the x86-64 binary.

**FUZZING RESULTS**

Two division-by-zero errors were found during fuzzing. Present in popularly used STM driver code, this has been fixed in their latest release.

**PERIPHERAL HANDLING EVALUATION**

Of the region identified as peripherals:

- 51% of the addresses were correctly identified
- Considering reserved regions, 88% are good to be fed with standard input data

**PRELIMINARY RETARGETING EVALUATION**

Rehosted applications compiled with ASAN were executed on Ubuntu 20.04 machine.

- The application tasks executed as expected without undefined crashes.
- Accesses to peripheral regions did not fail.
- Execution proceeded as input was provided via standard input