Contribution ID: **168** Type: **not specified**

Verifying the Conformance of a VirtIO Driver to the VirtIO Specification

Friday, 20 September 2024 15:15 (30 minutes)

VirtIO is a specification for virtual devices that describes how devices and drivers are defined and how they interact. For example, the specification defines the steps that a driver has to follow to initialize a virtio-device. The specification defines what is expected from a driver when communicating with a virtio-device. This specification has been used to develop virtio-drivers and virtio-devices in different languages and technologies. For example, QEMU implements virtio-devices in C. Rust-vmm implements virtio-devices in Rust. Recently, the specification has been used to build virtio-devices in hardware. Also, there are different implementations of the drivers depending on the operating system, e.g, Linux, Windows or baremetal. To ensure compatibility between different implementations, developers must ensure that the implementation conforms to the VirtIO specification. This is a manual task that relies on testing the implementation during different use cases. In this talk, we present a proof-of-concept solution that aims to systematically verify that a virtio-driver conforms to the VirtIO specification. During this exploratory work, we focus on a small section of the specification, which is the device lifecycle VIRTIO_CONFIG_S_* status register state machine. This section specifies the steps that a virtio-driver has to follow to initialize a virtio-device. We propose to encode these steps by using the Clock Constraint Specification Language (CCSL). This is a formal language that allows expressing the specification in terms of events and timing relationships between these events, e.g, causality. Then, we use this specification to check whether a virtio-driver follows the VirtIO specification. To do this, we use the ftrace interface to observe the execution of the virtio-driver. We apply our approach the traditional virtio memory balloon device to manage guest memory. During the initialization of the driver, violations to the specification are immediately informed to the user on the dmesg console. The aim of this talk is to present the approach and to have face-to-face discussions and debate about the benefits, drawbacks and trade-offs. We report take away lessons and present the tools to get the community familiar with the workflow.

Primary author: VARA LARSEN, Matias (Redhat)

Presenter: VARA LARSEN, Matias (Redhat)

Session Classification: Safe Systems with Linux MC

Track Classification: Safe Systems with Linux MC