Contribution ID: **51** Type: **not specified**

Power Management and USB Fuzzing: A Modular Approach to Kernel Coverage During Suspend Crashes

Saturday 13 December 2025 17:45 (45 minutes)

In Linux, testing the power management (PM) subsystem, particularly during suspend crashes, presents unique challenges. Logging mechanisms are often suspended during these events, making it difficult to capture critical information about system behavior. To address this gap, we developed a fuzzing system that combines LibAFL for fuzzing, S2E for external basic block coverage, and an input processor to synchronize and manage events across a virtual machine (VM) and an external board emulating a USB keyboard device passed through to the VM.

Unlike existing USB fuzzing tools like Syzkaller, which focus primarily on the enumeration phase, our system provides a persistent USB connection and allows deep coverage of the kernel during periods when self-coverage is unavailable. While this setup is currently focused on discovering races related to power management events such as suspend, it is designed with modularity in mind. Making it extendable for other functionality or fuzzing other subsystems where a persistent USB device is beneficial.

In development, the first challenge we encountered was ensuring coverage information could be safeguarded during suspend crashes. We initially explored solutions like kernel-level logging and QEMU's QMP socket-based communication, but these approaches failed as they lost functionality during suspend. We also considered using the Real-Time Clock (RTC) value to retain coverage information, as this was thought to be the only data recoverable. However, this method did not work in a virtual machine (VM) with KVM, where the RTC value is not persistent as it is on a real host.

To address this, we turned to external tools capable of extracting coverage from a running VM kernel. After evaluating options such as Unicorn and kAFL, we chose S2E due to its extensive documentation, ease of modification with plugins, and its support for basic block coverage. S2E allowed us to track kernel execution flow during suspend crashes, enabling us to gather coverage of the kernel even when the system could not self-monitor.

We then configured an S2E project to run a custom kernel image that supports power management events, USB passthrough, and remote suspension commands via SSH. A custom S2E plugin was developed to monitor user-specified program counter addresses and return a coverage map. This map informs the mutator about the executed program counters, helping determine the next sequence of events to generate.

For the fuzzing component, we evaluated multiple options. We first considered using uEmu, which provides a fuzzing plugin for S2E. However, we found it to be quite specific to certain use cases. Instead, we opted for LibAFL due to its flexibility and ease of integration with our modular fuzzing setup, which also facilitates future modifications to the system. Currently, we are using the mutator to generate a simple byte array that the system interprets as different events, such as USB keypresses and suspend events.

To trigger these events and ensure they are properly executed within the system, another process, the input processor is used. The input processor is responsible for taking the generated input sequence and distributing it appropriately across the VM and the external USB device emulator, connecting to both systems through SSH. Additionally, the S2E plugin and the LibAFL processor utilize shared status flags, allowing events to be synchronized with the input processor.

For the USB device emulation, we chose the ROCKPI 4B board due to its support for USB 3.0 On-the-Go (OTG), enabling the board to act as both a host and a device. The device is emulated using the raw-gadget

framework, with modifications to its keyboard example. Although this setup reliably sends keypresses and a remote wakeup signal via a keypress, the remote wakeup from USB is not currently functional.

The remote wakeup issue mainly involves QEMU's limited support for advanced power management functionalities since they are normally not needed. This results in missing ACPI tables and an emulated XHCI hub without wakeup functionality. QEMU could potentially be modified to support this, but we currently work around this by using the QEMU QMP socket shell, even though this bypasses the USB channels for proper coverage of a remote wakeup event. This is currently the primary limitation of the setup for fuzzing PM events.

Our framework includes a Coverage Tool to configure the coverage guidance of the fuzzing campaign for a specific part of the kernel. The Coverage Tool is a Java-based static analysis tool designed to analyze XML-format representations of source code modules and generate relationships between functions within a given module. Given a directory of XML files the tool extracts information including caller/callee relationships, line range coverage, and callback associations.

Future work on this solution will focus on resolving the remote wakeup issue and expanding the fuzzing setup with more configurable options. Such as specifying suspend timing (e.g., how long to wait after a keypress before suspending) and extending the system to fuzz additional subsystems beyond power management. Additionally, we would like to test different USB devices, such as a mass storage device, and explore if using a fully emulated device on the host at the file system level is possible instead of a physical board.

This work presents a unique fuzzing setup that uses a persistent USB keyboard for deep kernel coverage during suspend crashes and other USB and power management interactions. The modular system integrates S2E for basic block coverage, LibAFL for fuzzing, and an input processor for event handling. This approach provides a valuable tool for uncovering and fixing USB-related bugs that are otherwise difficult to cover, especially during power management events. This solution can also be extended to test other subsystems, devices, and features in the future due to its modular design.

Primary authors: Mr RAMOS, DARRION (University of Florida); Ms SIVER, VICTORIA (University of Florida); Dr BAI, Ken Yihang (University of Florida); Dr YAVUZ, Tuba (University of Florida)

Presenters: Mr RAMOS, DARRION (University of Florida); Ms SIVER, VICTORIA (University of Florida); Dr BAI, Ken Yihang (University of Florida); Dr YAVUZ, Tuba (University of Florida)

Session Classification: LPC Refereed Track

Track Classification: LPC Refereed Track