Contribution ID: **227** Type: **not specified**

PID FDs: where we were, where we are and were we would like to go

Thursday, 19 September 2024 12:00 (25 minutes)

Process ID File Descriptors were introduced in Linux v5.3. They allow tracking a process reliably, without risking races and reuse attacks, as they always refer to one single process regardless of the actual PID, so if the process goes away the file descriptor will become invalid, even if a new process with the same PID reappears at the same time.

Recently work has been done to plumb PID FDs through low-level userspace components - glibc returns the child's PID FD on pidfd_spawn(), systemd tracks processes via PID FDs and is able to receive queries asking for the session information or unit information via a PID FD, D-Bus implementations return the PID FD of a D-Bus endpoint via GetConnectionCredentials()/GetConnectionUnixProcessFD() (and they track processes via FD rather than PID), and Polkit allows writing rules authorizing by the systemd service name, which is possible to do safely thanks to using FDs all the way through. And now there is a new in-kernel pseudo-filesystem that assigns a unique ID to each PID FD, that never wraps.

We'll quickly summarize what we have and what use cases PID FDs have made possible, and then we'll move on to talk about what the next steps should ideally be in order to further enhance the feature and provide more functionality for userspace.

Primary author: Mr BOCCASSI, Luca (Microsoft)

Presenter: Mr BOCCASSI, Luca (Microsoft)

Session Classification: Kernel <-> Userspace/Init/System Management boundaries and APIs MC

Track Classification: Kernel <-> Userspace/Init/System Management boundaries and APIs MC