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

# eBPF-based tracing tools under 32 bit architectures

*Thursday, November 15, 2018 11:40 AM (20 minutes)*

Complex software usually depends on many different components, which sometimes perform background tasks with side effects not directly visible to their users. Without proper tools it can be hard to identify which component is responsible for performance hits or undesired behaviors.

We were challenged to implement D-Bus observability tools in embedded, ARM32 or ARM64 kernel based environments, both with 32-bit userspace. While we found bcc-tools, an open source compiler set useful, it appeared that it lacks support for 32-bit environments. We extended bcc-tools with support for 32-bit architectures. Using bcc-tools we created Linux eBPF programs – small programs written in a subset of C language, loaded from user-space and executed in kernel context. We attached them to uprobes and kprobes - user and kernel space special kinds of breakpoints. While it worked on ARM32 kernel based system, we faced another problem - ARM64 kernel lacked support for uprobes set in 32-bit binaries. The 64-bit ARM Linux kernel was extended with the ability to probe 32-bit binaries.

We propose to discuss challenges we faced trying to implement bcc-tools based tracing tools on ARM devices. We present a working solution to overcome lack of support for 32-bit architectures in bcc-tools, leaving space for discussion about other ways to achieve the same result. We also introduce 32-bit instruction probing in ARM64 kernel - a solution that we found very useful in our case. As a proof of concept we present tools that monitor D-Bus usage in ARM32 or ARM64 kernel based system with 32-bit userspace. We list what needs to be done for complete eBPF-based tools to be fully usable on ARM.

**Presenters:** SLODCZYK, Maciej (Samsung); SZYNDELA, Adrian (Samsung)

**Session Classification:** BPF MC