Contribution ID: **173** Type: **not specified**

Augmenting syscalls in 'perf trace' using eBPF

Thursday, 15 November 2018 14:40 (20 minutes)

The 'perf trace' tool uses the syscall tracepoints to provide a !ptrace based 'strace' like tool, augmenting the syscall arguments provided by the tracepoints with integer->strings tables automatically generated from the kernel headers, showing the paths associated with fds, pid COMMs, etc.

That is enough for integer arguments, pointer arguments needs either kprobes put in special locations, which is fragile and has been so far implemented only for getname_flags (open, etc filenames), or using eBPF to hook into the syscall enter/exit tracepoints to collect pointer contents right after the existing tracepoint payload.

This has been done to some extent and is present in the kernel sources in the tools/perf/examples/bpf/augmented_syscalls.c, using the pre-existing support in perf to use BPF C programs as event names, automagically using clang/llvm to build and load it via sys_bpf(), 'perf trace' hooks this to the existing beautifiers that seeing that extra data use it to get the filename, struct sockaddr_in, etc.

This was done for a bunch of syscalls, what is left is to get this all automated using BTF, allow passing filters attached to the syscalls, select which syscalls should be traced, use a pre-compiled augmented_syscalls.c just selecting what bits of the obj should be used, etc, i.e. the open issues about this streamlining process to avoid requiring the clang toolchain, etc will be the matter of this discussion.

I agree to abide by the anti-harassment policy

Presenter: MELO, Arnaldo Carvalho de (Red Hat)

Session Classification: BPF MC