perf trace

- raw_syscalls:sys_enter and sys_exit
- strace like
- perf targets: system wide, CPU, cgroups, etc
- Much lower overhead than strace
- Mix and match with other tracepoints, kprobes, uprobes, etc
- But only integer args

Syscall pointer args: 1st "solution"

- kprobes at strategic locations
- getname_flags() for filenames
- Needs to combine syscall event with kprobes one
- fragile: probe location changes over time

Augmented Syscalls: eBPF

- Hooks into raw_syscalls:sys_enter
- Appends pointer contents to existing payload
- Existing beautifiers changed to use it
- When available, continue working without it
- No need for probe:vfs_getname
- Filters
- Extend the kernel without writing "kernel" code

Augmented Syscalls

- bpf-output event
- bpf_perf_event_output
- bpf_probe_read
- bpf_probe_read_str

Augmenting syscalls

```
struct bpf_map SEC("maps") __augmented__ = {
       .type = BPF_MAP_TYPE_PERF_EVENT_ARRAY,
}
int syscall_enter(syscall)(struct syscall_enter_open_args *args)
{
        struct augmented_open_args augmented;
        probe_read(&augmented.args, sizeof(augmented.args), args);
        str_len = probe_read_str(&augmented.filename.value,
                                 sizeof(augmented.filename.value),
                                 args->filename_ptr);
        perf_event_output(args, &__augmented__, BPF_F_CURRENT_CPU,
                          &augmented, sizeof(augmented));
        return 0;
}
```

Augmented Syscalls

```
# cd tools/perf/examples/bpf/
# perf trace -e augmented_syscalls.c cat hello.c
#include <stdio.h>

int syscall_enter(openat)(void *args)
{
        puts("Hello, world\n");
        return 0;
}

license(GPL);
 0.000 cat/31285 openat(dfd: CWD, filename: /etc/ld.so.cache, flags: C
 0.029 cat/31285 openat(dfd: CWD, filename: /lib64/libc.so.6, flags: C
 0.250 cat/31285 open(filename: /usr/lib/locale/locale-archive, flags:
 0.300 cat/31285 openat(dfd: CWD, filename: hello.c)
#
```

Further details: maps

- syscall numbers loaded from userspace
- syscall arg types in another map
- together with how many bytes to copy
- types: kernel BTF section
- syscalls to trace: in a map too (done)
- One kernel .o

Difficulties: real or feared

- clang optimizations versus validator
- size constraints
- string operations

Difficulties: real

- barriers to avoid combining ctx accesses
- size constraints: improve libbpf/verifier messages
- clang-tidy

TODO

- BTF struct auto-beautifier in 'perf trace'
- mapping of --filter-pids to BPF (done)
- syscall arg filters (should be done)
- Merge Jiri Olsa's 'perf bpf'
- Userspace/stap VM, access to perf userspace functions