Summary for Winter Holiday

Haonan Li

COMPASS

Feb 25, 2021

1 Outline

1 Recall: why we need hook syscall?

- In replay stage, we cannot re-construct the return value of **syscall**
- Find a way to log all **syscalls** in the record stage

```
static void el0_svc_common(struct pt_regs *regs, int scno, int sc_nr,
                           const syscall_fn_t syscall_table[])
{
  // ... some pre-check ...

    invoke_syscall(regs, scno, sc_nr, syscall_table);

  // ... tracing status check

trace_exit:
    syscall_trace_exit(regs);
}
```

Linux has provided hook positions.

```
void syscall_trace_exit(struct pt_regs *regs)
{
    audit_syscall_exit(regs);

    if (test_thread_flag(TIF_SYSCALL_TRACEPOINT))
        trace_sys_exit(regs, regs_return_value(regs));

    if (test_thread_flag(TIF_SYSCALL_TRACE))
        tracehook_report_syscall(regs, PTRACE_SYSCALL_EXIT);

    rseq_syscall(regs);
}
```

- **sysdig** is a universal system visibility tool.
- sysdig leverages `tracepoints` and load drivers to capture kernel events.

A receipt for using sysdig to capture syscalls for a process named zsh and with pid 3981

```
root@ubuntu:/home# sysdig proc.name=zsh and proc.pid=3981
3344 ... < read res=1 data=z
3345 ... > rt_sigprocmask
3346 ... < rt_sigprocmask
3349 ... > fcntl fd=0(<f>/dev/pts/2) cmd=1(F_DUPFD)
3350 ... < fcntl res=11(<f>/dev/pts/2)
3351 ... > close fd=0(<f>/dev/pts/2)
3352 ... < close res=0
3353 ... > openat
3357 ... < openat fd=0(<f>/dev/null) dirfd=-100(AT_FDCWD) ...
3362 ... > mmap addr=0 length=16384 prot=3(PROT_READ|PROT_WRITE) ...
3363 ... < mmap res=7F3CB2E39000 vm_size=53820 vm_rss=6456 vm_swap=0
3364 ... > rt_sigprocmask
3365 ... < rt_sigprocmask
```

- We need install sysdig from source code and configure all dependencies manually.
- Or we could also hook syscalls as sysdig does.

We need do some experiments to:

- figure out the overhead.
- judge whether the information captured is enough.
- make a demo to finish original schedule.