What Is Still Missing in System Call Tracing

Renzo Davoli Davide Berardi

FOSDEM 2026: February 1st, 2026

Kernel Devroom

This work was partially supported by the project SERICS (PE00000014) under the NRRP MUR program funded by the European Union - NextGenerationEU.

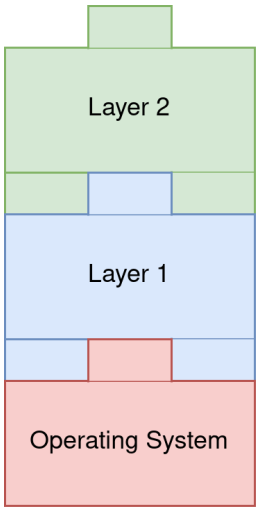
Mainly for debug features system calls can be hijacked to print information on which systemcall has been executed:
An example of this feature is `strace`, a program that prints every systemcall executed by another program.

```
debian@purelibc-injector:~/syscall_tracing/ptrace$ strace cat /etc/passwd 2>&1 | grep read
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\20t\2\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0"..., 784, 64) = 784
read(3, "# Locale name alias data base.\n#"..., 4096) = 2996
read(3, "", 4096)                       = 0
read(3, "root:x:0:0:root:/root:/bin/bash\n"..., 131072) = 1351
read(3, "", 131072)                     = 0
```

https://github.com/virtualsquare/syscall_tracing

System calls are an interface to virtualize the system creating a virtualization similar to os-level one (i.e. namespaces).

With this feature is possible to create the concept of MLOS: Multiple Layer Operating System. Every layer has the same interface of the other (systemcalls, e.g. POSIX?)



https://github.com/virtualsquare/syscall_tracing

Systemcalls can be hijacked by instructing the kernel to tell to an userspace program (hypervisor) which systemcall has been invoked and with which parameters. We will refer to this technique as "hypervisor mode".

On the other side programs can declare to be traced by themselves (e.g. by changing the function called when executing a specific systemcall-wrapper, we refer to this as "self-virtualization".

`https://github.com/virtualsquare/syscall_tracing`

Hypervisors

PSSI: PTRACE_SET_SYSCALL_INFO

Backend	avail	skip	event	arch indep
ptrace	Yes	No	Wait+ptrace	No
ptrace+seccomp	Yes	Yes	Wait+ptrace	No
ptrace+seccomp+PSSI	Yes	No	Wait+ptrace	Almost
ptrace+seccomp+PSSI+patch	No[1]	Yes	Wait+ptrace	Yes
seccomp_unotify	Yes [2]	Yes[3]	Poll+ioctl	Yes
seccomp_unotify+patch	No [2] [4]	Yes[3]	Poll+ioctl	Yes

Self Virtualization

Backend	avail	nested	event	arch indep
purelibc	Yes [5]	Yes	function call	Yes
pr_set_syscall_user_dispatch	Yes [2]	hard	SIGSYS+ucontext	No

[1] patch:add ptrace_syscall_info_seccomp_skip+ptrace_syscall_info_flag_set_ip

[2] it needs a trick to trace all the syscalls

[3] no way to retrieve the result value of *real* syscalls

[4] patch:add seccomp_unotify_flag_fdupfd+seccomp_unotify_closefd

[5] incomplete support

`https://github.com/virtualsquare/syscall_tracing`

Patch 01: Actually, it is not possible to skip the system call using nr = -1 at the PTRACE_SYSCALL_INFO_SECCOMP phase as it is no way to set the return value. This patch introduces this feature using the new PTRACE_SYSCALL_INFO_SECCOMP_SKIP tag.

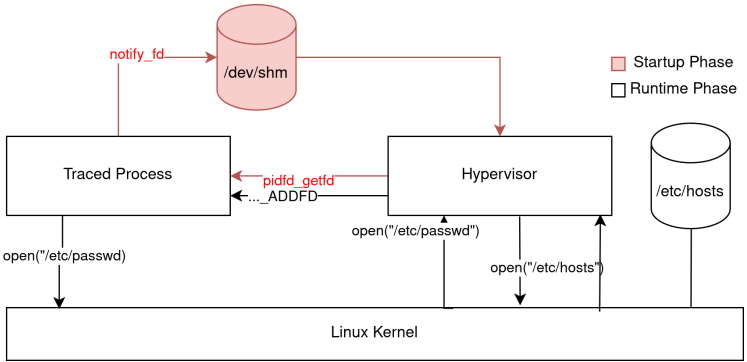
Patch 03: In the kernel the instruction_pointer_set function is already available in almost all the architectures. This patch introduces the missing functions for the other architectures.

Patch 04: As in 01, it is not possible to set the instruction pointer using ptrace. This patch introduces the new features using the new flag PTRACE_SYSCALL_INFO_FLAG_SET_IP.

Acknowledgement: Dmitry V. Levin.

`https://github.com/virtualsquare/syscall_tracing`

With seccomp unotify it is impossible to implement tracing of system call return values when the system call has to been executed in the tracee. For this reason, the system calls must be executed inside the tracer, and the return values must be passed back (including kernel handles such as file descriptors).



https://github.com/virtualsquare/syscall_tracing

Seccomp unotify provides the ability to send file descriptors, either by automatically selecting the first available file descriptor number or by using the file descriptor specified in the call. If this specified file descriptor refers to a file that is already open, it is closed first (similarly to the behavior of dup2).

Operations such as fcntl with the F_DUPFD flag cannot be implemented with seccomp unotify. This flag indicates that the first available file descriptor starting from a given value should be used (e.g., use the first file descriptor greater than or equal to 100).

This patch also allows the tracer to close a file that is currently open in the tracee. This functionality is necessary because it is not possible to obtain the return value of a close system call executed in the tracee (due to atomicity issues of the operations).

`https://github.com/virtualsquare/syscall_tracing`

The repository includes the following sources:

ptrace_set_syscall_info_patches Kernel patches to implement missing features in PTRACE_SET_SYSCALL_INFO.

ptrace_set_syscall_info_tracer_almost_arch_independent Proof-of-concept to trace system calls using PTRACE_SET_SYSCALL_INFO.

seccomp_unotify_patch Kernel patch to implement two missing features in seccomp unotify.

syscall_unotify_tracer Proof-of-concept to trace system calls using seccomp unotify.

---

https://github.com/virtualsquare/syscall_tracing

Thank you for your attention!

renzo@cs.unibo.it

davide.berardi@unimercatorum.it