Syscall User Dispatch

Background

Compatibility layers like Wine need a way to efficiently emulate system calls of only a part of their process - the part that has the incompatible code - while being able to execute native syscalls without a high performance penalty on the native part of the process. Seccomp falls short on this task, since it has limited support to efficiently filter syscalls based on memory regions, and it doesn't support removing filters. Therefore a new mechanism is necessary.

Syscall User Dispatch brings the filtering of the syscall dispatcher address back to userspace. The application is in control of a flip switch, indicating the current personality of the process. A multiple-personality application can then flip the switch without invoking the kernel, when crossing the compatibility layer API boundaries, to enable/disable the syscall redirection and execute syscalls directly (disabled) or send them to be emulated in userspace through a SIGSYS.

The goal of this design is to provide very quick compatibility layer boundary crosses, which is achieved by not executing a syscall to change personality every time the compatibility layer executes. Instead, a userspace memory region exposed to the kernel indicates the current personality, and the application simply modifies that variable to configure the mechanism.

There is a relatively high cost associated with handling signals on most architectures, like x86, but at least for Wine, syscalls issued by native Windows code are currently not known to be a performance problem, since they are quite rare, at least for modern gaming applications.

Since this mechanism is designed to capture syscalls issued by non-native applications, it must function on syscalls whose invocation ABI is completely unexpected to Linux. Syscall User Dispatch, therefore doesn't rely on any of the syscall ABI to make the filtering. It uses only the syscall dispatcher address and the userspace key.

As the ABI of these intercepted syscalls is unknown to Linux, these syscalls are not instrumentable via ptrace or the syscall tracepoints.

Interface

A thread can setup this mechanism on supported kernels by executing the following prctl:

> prctl(PR_SET_SYSCALL_USER_DISPATCH, <op>, <offset>, <length>, [selector])

<op> is either PR_SYS_DISPATCH_ON or PR_SYS_DISPATCH_OFF, to enable and disable the mechanism globally for that thread. When PR_SYS_DISPATCH_OFF is used, the other fields must be zero.

[<offset>, <offset>+<length>) delimit a memory region interval from which syscalls are always executed directly, regardless of the userspace selector. This provides a fast path for the C library, which includes the most common syscall dispatchers in the native code applications, and also provides a way for the signal handler to return without triggering a nested SIGSYS on (rt_)sigreturn. Users of this interface should make sure that at least the signal trampoline code is included in this region. In addition, for syscalls that implement the trampoline code on the vDSO, that trampoline is never intercepted.

[selector] is a pointer to a char-sized region in the process memory region, that provides a quick way to enable disable syscall redirection thread-wide, without the need to invoke the kernel directly. selector can be set to SYSCALL_DISPATCH_FILTER_ALLOW or SYSCALL_DISPATCH_FILTER_BLOCK. Any other value should terminate the program with a SIGSYS.

Security Notes

Syscall User Dispatch provides functionality for compatibility layers to quickly capture system calls issued by a non-native part of the application, while not impacting the Linux native regions of the process. It is not a mechanism for sandboxing system calls, and it should not be seen as a security mechanism, since it is trivial for a malicious application to subvert the mechanism by jumping to an allowed dispatcher region prior to executing the syscall, or to discover the address and modify the selector value. If the use case requires any kind of security sandboxing, Seccomp should be used instead.

Any fork or exec of the existing process resets the mechanism to PR_SYS_DISPATCH_OFF.