No New Privileges Flag

The execve system call can grant a newly-started program privileges that its parent did not have. The most obvious examples are setuid/setgid programs and file capabilities. To prevent the parent program from gaining these privileges as well, the kernel and user code must be careful to prevent the parent from doing anything that could subvert the child. For example:

- The dynamic loader handles `LD_*` environment variables differently if a program is setuid.
- chroot is disallowed to unprivileged processes, since it would allow `/etc/passwd` to be replaced from the point of view of a process that inherited chroot.
- The exec code has special handling for ptrace.

These are all ad-hoc fixes. The `no_new_privs` bit (since Linux 3.5) is a new, generic mechanism to make it safe for a process to modify its execution environment in a manner that persists across execve. Any task can set `no_new_privs`. Once the bit is set, it is inherited across fork, clone, and execve and cannot be unset. With `no_new_privs` set, `execve()` promises not to grant the privilege to do anything that could not have been done without the execve call. For example, the setuid and setgid bits will no longer change the uid or gid; file capabilities will not add to the permitted set, and LSMs will not relax constraints after execve.

To set `no_new_privs`, use:

```
prctl(PR_SET_NO_NEW_PRIVS, 1, 0, 0, 0);
```

Be careful, though: LSMs might also not tighten constraints on exec in `no_new_privs` mode. (This means that setting up a general-purpose service launcher to set `no_new_privs` before execing daemons may interfere with LSM-based sandboxing.)

Note that `no_new_privs` does not prevent privilege changes that do not involve `execve()`. An appropriately privileged task can still call `setuid(2)` and receive SCM_RIGHTS datagrams.

There are two main use cases for `no_new_privs` so far:

- Filters installed for the seccomp mode 2 sandbox persist across execve and can change the behavior of newly-executed programs. Unprivileged users are therefore only allowed to install such filters if `no_new_privs` is set.
- By itself, `no_new_privs` can be used to reduce the attack surface available to an unprivileged user. If everything running with a given uid has `no_new_privs` set, then that uid will be unable to escalate its privileges by directly attacking setuid, setgid, and fcap-using binaries; it will need to compromise something without the `no_new_privs` bit set first.

In the future, other potentially dangerous kernel features could become available to unprivileged tasks if `no_new_privs` is set. In principle, several options to `unshare(2)` and `clone(2)` would be safe when `no_new_privs` is set, and `no_new_privs` + `chroot` is considerable less dangerous than chroot by itself.