Yama is a Linux Security Module that collects system-wide DAC security
protections that are not handled by the core kernel itself. This is
selectable at build-time with CONFIG_SECURITY_YAMA, and can be controlled
at run-time through sysctls in /proc/sys/kernel/yama:

- ptrace_scope

==

ptrace_scope:

As Linux grows in popularity, it will become a larger target for
malware. One particularly troubling weakness of the Linux process
interfaces is that a single user is able to examine the memory and
running state of any of their processes. For example, if one application
(e.g. Pidgin) was compromised, it would be possible for an attacker to
attach to other running processes (e.g. Firefox, SSH sessions, GPG agent,
etc) to extract additional credentials and continue to expand the scope
of their attack without resorting to user-assisted phishing.

This is not a theoretical problem. SSH session hijacking
(http://www.storm.net.nz/projects/7) and arbitrary code injection
(http://c-skills.blogspot.com/2007/05/injectso.html) attacks already
exist and remain possible if ptrace is allowed to operate as before.
Since ptrace is not commonly used by non-developers and non-admins, system
builders should be allowed the option to disable this debugging system.

For a solution, some applications use prctl(PR_SET_DUMPABLE, ...) to
specifically disallow such ptrace attachment (e.g. ssh-agent), but many
do not. A more general solution is to only allow ptrace directly from a
parent to a child process (i.e. direct "gdb EXE" and "strace EXE" still
work), or with CAP_SYS_PTRACE (i.e. "gdb --pid=PID", and "strace -p PID"
still work as root).

In mode 1, software that has defined application-specific relationships
between a debugging process and its inferior (crash handlers, etc),
prctl(PR_SET_PTRACER, pid, ...) can be used. An inferior can declare which
other process (and its descendants) are allowed to call PTRACE_ATTACH
against it. Only one such declared debugging process can exists for
each inferior at a time. For example, this is used by KDE, Chromium, and
Firefox's crash handlers, and by Wine for allowing only Wine processes
to ptrace each other. If a process wishes to entirely disable these ptrace
restrictions, it can call prctl(PR_SET_PTRACER, PR_SET_PTRACER_ANY, ...)
so that any otherwise allowed process (even those in external pid namespaces)
may attach.

The sysctl settings (writable only with CAP_SYS_PTRACE) are:

0 - classic ptrace permissions: a process can PTRACE_ATTACH to any other
    process running under the same uid, as long as it is dumpable (i.e.
    did not transition uids, start privileged, or have called
    prctl(PR_SET_DUMPABLE...) already). Similarly, PTRACE_TRACEME is
    unchanged.

1 - restricted ptrace: a process must have a predefined relationship
    with the inferior it wants to call PTRACE_ATTACH on. By default,
    this relationship is that of only its descendants when the above
    classic criteria is also met. To change the relationship, an
    inferior can call prctl(PR_SET_PTRACER, debugger, ...) to declare
    an allowed debugger PID to call PTRACE_ATTACH on the inferior.
    Using PTRACE_TRACEME is unchanged.

2 - admin-only attach: only processes with CAP_SYS_PTRACE may use ptrace
    with PTRACE_ATTACH, or through children calling PTRACE_TRACEME.

```
3 - no attach: no processes may use ptrace with PTRACE_ATTACH nor via
    PTRACE_TRACEME. Once set, this sysctl value cannot be changed.

The original children-only logic was based on the restrictions in grsecurity.

==============================================================
```