Netlink engine issues, and ways to fix those

Andrei Vagin, Linux Kernel Summit, 2016

Limitations of /proc/PID interface

- Requires *at least* three syscalls per process per file
  - open(), read(), close()
- Variety of formats, mostly text based
- Some formats are non-extendable
  - /proc/PID/maps where the last column is optional
- Sometimes slow due to extra attributes
  - /proc/PID/smaps vs /proc/PID/maps

/proc/PID/smaps

```
7f1cb0afc000-7f1cb0afd000 rw-p 00021000 08:03 656516    /usr/lib64/ld-2.21.so
Size:                  4 kB
Rss:                   4 kB
Pss:                   4 kB
Shared_Clean:          0 kB
Shared_Dirty:          0 kB
Private_Clean:         0 kB
Private_Dirty:         4 kB
Referenced:            4 kB
Anonymous:             4 kB
AnonHugePages:         0 kB
Swap:                  0 kB
KernelPageSize:        4 kB
MMUPageSize:           4 kB
Locked:                0 kB
VmFlags: rd wr mr mw me dw ac sd
```

```
$ time cat /proc/*/maps > /dev/null
real   0m0.061s
user   0m0.002s
sys    0m0.059s
```

```
$ time cat /proc/*/smaps > /dev/null
real   0m0.253s
user   0m0.004s
sys    0m0.247s
```

Procfs (perf output)

```
  Children      Self  Command        Shared Object     Symbol
-   97.98%     0.24%  task_proc_all  [kernel.vmlinux]  [k] entry_SYSCALL_64_fastpath
   - 97.74% entry_SYSCALL_64_fastpath
      - 68.20% sys_read
         - 67.94% vfs_read
            - 67.66% __vfs_read
               - 67.08% seq_read
                  - 66.51% proc_single_show
                     - 65.47% proc_pid_status
                        - 20.81% render_sigset_t
                           - 18.27% seq_printf
                              + 15.77% seq_vprintf
                        - 10.65% task_mem
                           + 8.78% seq_print
                          + 1.02% hugetlb_report_usage
                        + 7.40% seq_printf
                        + 5.12% cpuset_task_status_allowed
                        + 4.67% render_cap_t
                        + 3.48% seq_put_decimal_ull
                       ....
      - 15.15% sys_getdents
           iterate_dir
         + proc_root_readdir
      - 12.07% sys_open
         - 11.94% do_sys_open
            + 10.73% do_filp_open
      + 1.90% syscall_return_slowpath
```

Benefits of task_diag (netlink-based interface)

- Expandable format
  - Can add new attributes or extend existing ones
- Customizable responses
  - Can specify what information is needed
- Binary format (no slow bin/text/bin conversion)
  - Bin to text (kernel); text to bin, bin to text (ps, top)
- File descriptor to save a state between syscalls
  - Can do several reads to get all the data

Task-diag (perf output)

```
  Children      Self  Command         Shared Object      Symbol
-   99.16%     0.04%  task_diag_all   [kernel.vmlinux]   [k] entry_SYSCALL_64_fastpath
   - 99.12% entry_SYSCALL_64_fastpath
      - 99.03% sys_read
         - vfs_read
            - 99.02% __vfs_read
                 proc_reg_read
               - task_diag_read
                  - 92.66% taskdiag_dumpit
                     + 68.85% next_tgid
                     + 4.09% ptrace_may_access
                       1.73% strncpy
                     + 1.56% get_task_comm
                       1.48% _raw_spin_lock
                     + 0.94% nla_reserve
                       0.72% __task_pid_nr_ns
                       0.67% task_tgid_nr_ns
                    5.95% task_tgid_nr_ns
```

Proc vs task_diag

- open(/proc/pid/loginuid), close()
  - sys 0m0.449s
- open(/proc/pid/loginuid), read(), close()
  - sys 0m0.477s
- open(/proc/pid/status), read(), close()
  - sys 0m2.001s
- Get process ids and credentials via task_diag
  - sys 0m0.179s

Task-diag (perf output)

syscall	calls	total (msec)	min (msec)	avg (msec)	max (msec)	stddev (%)
read	53	212.103	0.003	4.002	12.077	10.50%
mmap	16	0.123	0.003	0.008	0.011	7.86%
mprotect	10	0.094	0.005	0.009	0.016	10.92%
open	7	0.068	0.005	0.010	0.026	28.99%
fstat	6	0.013	0.002	0.002	0.002	3.84%
close	6	0.012	0.002	0.002	0.002	2.07%
munmap	1	0.011	0.011	0.011	0.011	0.00%
brk	3	0.011	0.002	0.004	0.006	29.22%
access	1	0.006	0.006	0.006	0.006	0.00%
write	1	0.005	0.005	0.005	0.005	0.00%
rt_sigaction	2	0.004	0.002	0.002	0.002	13.00%
getrlimit	1	0.002	0.002	0.002	0.002	0.00%
rt_sigprocmask	1	0.002	0.002	0.002	0.002	0.00%
arch_prctl	1	0.002	0.002	0.002	0.002	0.00%
set_tid_address	1	0.002	0.002	0.002	0.002	0.00%
set_robust_list	1	0.002	0.002	0.002	0.002	0.00%

Task-diag (perf output)

syscall	calls	total (msec)	min (msec)	avg (msec)	max (msec)	stddev (%)
read	100372	1455.428	0.003	0.015	0.045	0.03%
open	100374	400.221	0.003	0.004	0.036	0.05%
getdents	97	319.779	0.001	3.297	7.449	5.51%
close	100374	209.038	0.001	0.002	0.014	0.08%
mmap	16	0.130	0.003	0.008	0.015	9.35%
mprotect	10	0.113	0.005	0.011	0.033	22.97%
fstat	8	0.021	0.002	0.003	0.004	8.46%
write	1	0.021	0.021	0.021	0.021	0.00%
munmap	1	0.011	0.011	0.011	0.011	0.00%
brk	3	0.011	0.002	0.004	0.006	28.56%
access	1	0.006	0.006	0.006	0.006	0.00%
rt_sigaction	2	0.004	0.002	0.002	0.002	10.00%
getrlimit	1	0.002	0.002	0.002	0.002	0.00%
rt_sigprocmask	1	0.002	0.002	0.002	0.002	0.00%
arch_prctl	1	0.002	0.002	0.002	0.002	0.00%
set_tid_address	1	0.002	0.002	0.002	0.002	0.00%
set_robust_list	1	0.002	0.002	0.002	0.002	0.00%

Issues of netlink by Andy Lutomirski

“I realize that it's convenient to use a socket as a context to keep state between syscalls, but it has some annoying side effects:

- It makes people want to rely on send()'s caller's creds.
- It's miserable in combination with seccomp.
- It doesn't play nicely with namespaces.
- It makes me wonder why things like task_diag, which have nothing to do with networking, seem to get tangled up with networking.“

A parallel interface by Andy Lutomirski

```
int issue_kernel_command(int ns, int command, const struct iovec *iov, int iovcnt, int flags);

ns is an actual namespace fd or:
    KERNEL_COMMAND_CURRENT_NETNS
    KERNEL_COMMAND_CURRENT_PIDNS
    Etc, or a special one:
    KERNEL_COMMAND_GLOBAL.  KERNEL_COMMAND_GLOBAL can't be used in a non-root namespace.

KERNEL_COMMAND_GLOBAL works even for namespaced things, if the relevant current ns is the init
namespace.  (This feature is optional, but it would allow gradually namespacing global things.)

command is an enumerated command.  Each command implies a namespace type, and, if you feed this
thing the wrong namespace type, you get EINVAL. The high bit of command indicates whether it's
read-only
command.

iov gives a command in the format expected, which, for the most part, would be a netlink
message.

The return value is an fd that you can call read/readv on to read the response.  It's not a
socket (or at least you can't do normal socket operations on it if it is a socket behind the
scenes).  The implementation of read() promises *not* to look at caller creds.  The returned fd
is unconditionally cloexec -- it's 2016 already.
```

Thank you!

Where is netlink used

- Net (route, firewall, socket-diag, nflog, xfrm)
- Taskstats
- Selinux
- Audit
- iSCSI
- Uevent
- Drivers

The netlink message format

```
struct nslmsg_hdr {
    nlmsg_len
    mlmsg_type
};
```

```
struct nlattr {
    nla_len
    nla_type
}
struct task_diag_base {
    tgid
    pid
    ppid
    ….
    comm
}
```

```
struct nlattr {
    nla_len
    nla_type
}
struct task_diag_creds {
    cap_inheritable
    cap_permitted
    ….
    fsgid
};
```

```
struct nlattr {
    nla_len
    nla_type
}
struct XXXX {
    ….
};
```