BPF_PROG_TYPE_CGROUP_SYSCTL

This document describes `BPF_PROG_TYPE_CGROUP_SYSCTL` program type that provides cgroup-bpf hook for sysctl.

The hook has to be attached to a cgroup and will be called every time a process inside that cgroup tries to read from or write to sysctl knob in proc.

1. Attach type

`BPF_CGROUP_SYSCTL` attach type has to be used to attach `BPF_PROG_TYPE_CGROUP_SYSCTL` program to a cgroup.

2. Context

`BPF_PROG_TYPE_CGROUP_SYSCTL` provides access to the following context from BPF program:

```
struct bpf_sysctl {
    __u32 write;
    __u32 file_pos;
};
```

- `write` indicates whether sysctl value is being read (`0`) or written (`1`). This field is read-only.
- `file_pos` indicates file position sysctl is being accessed at, read or written. This field is read-write. Writing to the field sets the starting position in sysctl proc file `read(2)` will be reading from or `write(2)` will be writing to. Writing zero to the field can be used e.g. to override whole sysctl value by `bpf_sysctl_set_new_value()` on `write(2)` even when it's called by user space on `file_pos > 0`. Writing non-zero value to the field can be used to access part of sysctl value starting from specified `file_pos`. Not all sysctl support access with `file_pos != 0`, e.g. writes to numeric sysctl entries must always be at file position `0`. See also `kernel.sysctl_writes_strict` sysctl.

See linux/bpf.h for more details on how context field can be accessed.

3. Return code

`BPF_PROG_TYPE_CGROUP_SYSCTL` program must return one of the following return codes:

- `0` means "reject access to sysctl";
- `1` means "proceed with access".

If program returns `0` user space will get `-1` from `read(2)` or `write(2)` and `errno` will be set to `EPERM`.

4. Helpers

Since sysctl knob is represented by a name and a value, sysctl specific BPF helpers focus on providing access to these properties:

- `bpf_sysctl_get_name()` to get sysctl name as it is visible in `/proc/sys` into provided by BPF program buffer;
- `bpf_sysctl_get_current_value()` to get string value currently held by sysctl into provided by BPF program buffer. This helper is available on both `read(2)` from and `write(2)` to sysctl;
- `bpf_sysctl_get_new_value()` to get new string value currently being written to sysctl before actual write happens. This helper can be used only on `ctx->write == 1`;
- `bpf_sysctl_set_new_value()` to override new string value currently being written to sysctl before actual write happens. Sysctl value will be overridden starting from the current `ctx->file_pos`. If the whole value has to be overridden BPF program can set `file_pos` to zero before calling to the helper. This helper can be used only on `ctx->write == 1`. New string value set by the helper is treated and verified by kernel same way as an equivalent string passed by user space.

BPF program sees sysctl value same way as user space does in proc filesystem, i.e. as a string. Since many sysctl values represent an integer or a vector of integers, the following helpers can be used to get numeric value from the string:

- `bpf_strtol()` to convert initial part of the string to long integer similar to user space strtol(3);
- `bpf_strtoul()` to convert initial part of the string to unsigned long integer similar to user space strtoul(3);

See linux/bpf.h for more details on helpers described here.

5. Examples

See test_sysctl_prog.c for an example of BPF program in C that access sysctl name and value, parses string value to get vector of integers and uses the result to make decision whether to allow or deny access to sysctl.

6. Notes

`BPF_PROG_TYPE_CGROUP_SYSCTL` is intended to be used in **trusted** root environment, for example to monitor sysctl usage or catch

unreasonable values an application, running as root in a separate cgroup, is trying to set.

Since *task_dfl_cgroup(current)* is called at *sys_read* / *sys_write* time it may return results different from that at *sys_open* time, i.e. process that opened sysctl file in proc filesystem may differ from process that is trying to read from / write to it and two such processes may run in different cgroups, what means `BPF_PROG_TYPE_CGROUP_SYSCTL` should not be used as a security mechanism to limit sysctl usage.

As with any cgroup-bpf program additional care should be taken if an application running as root in a cgroup should not be allowed to detach/replace BPF program attached by administrator.