



CONTAINERS AND CHECKPOINT/RESTORE MC

# Checkpoint/Restore of Device Cgroup eBPF Programs

Pavel Tikhomirov · Virtuozzo



LINUX  
PLUMBERS  
CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

## CONTEXT

# Device access in containers became eBPF

## cgroup v2

Virtuozzo containers switched to the cgroup v2 unified hierarchy, and we want feature parity with v1.

## No devices.allow

There is no easy-to-dump devices.list. Access control is a **CGROUP\_DEVICE** BPF program type attached to the cgroup.

## Nested Docker

runc inside Virtuozzo container wants to load and attach these cgroup BPF programs. The mainline kernel forbids that in a container. Virtuozzo allows it.

Container carries BPF programs of nested CTs, so we have to preserve them in CRIU, but we have no API to save them.



LINUX  
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



# BPF tokens and VE\_FEATURE\_BPF

## Bpffs tokens

allow load (bpf\_token\_capable())

```
delegate_cmds=prog_load  
delegate_progs=cgroup_device  
delegate_attachs=cgroup_device
```

app should know how to use a token  
(AFAIK runc does not support tokens)

## Virtuozzo specific way

per VE:

- allow load (BPF\_PROG\_LOAD restricted same as unprivileged\_bpf\_disabled = false)
- allow query (BPF\_PROG\_QUERY restricted to own cgroup subtree)
- add program limit

**Tokens already let containers load programs, so migrating such a container will need the programs back in a reloadable form.**





## THE PROBLEM

**The kernel accepts a program, but gives no way to get it back in a reloadable form.**

**BPF\_PROG\_LOAD:** original insns in

**BPF\_OBJ\_GET\_INFO\_BY\_FD:** xlated or JITed out



**LINUX  
PLUMBERS  
CONFERENCE**

PRAGUE, CZECHIA  
OCT. 5-7, 2026

## WHY NOT DUMP WHAT THE KERNEL HAS?

# Xlated is constant-blinded

With `bpf_jit_harden`  $\geq 1$ , every constant in the program is hidden behind random values, unless the loader is `bpf_capable()`. Inside a VE it never is.

```
orig:    if r2 != 1
xlated:  r11 = rnd;
         r11 ^= rnd^1;
         if r2 != r11
```

Reload fails: **R11 is invalid**. Seen on every `systemd` and `runc` `devcg` program in a CT.

## Also kernel-specific

- jited: native code, not loadable at all.
- Both xlated/jitted dumps need `bpf_dump_raw_ok()`.

**The original copy solves this:** it is saved before blinding. De-blinding xlated instead would depend on kernel rewrite patterns that are not ABI.



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



## PRIOR ART

# Seccomp solved this a decade ago

Seccomp filters are translated at attach time too. With `CONFIG_CHECKPOINT_RESTORE` the kernel keeps the original filter, and CRIU reads it back to checkpoint seccomp. We do the same for eBPF.

|              | seccomp (since Linux 4.4) | devcg eBPF (this work)           |
|--------------|---------------------------|----------------------------------|
| Kept copy    | fp->orig_prog             | prog->aux->orig_insns            |
| Gated on     | CONFIG_CHECKPOINT_RESTORE | CONFIG_CHECKPOINT_RESTORE        |
| Read back by | PTRACE_SECCOMP_GET_FILTER | BPF_OBJ_GET_INFO_BY_FD           |
| Restored by  | re-installing the filter  | BPF_PROG_LOAD: re-verify, re-JIT |



# Keep the original instructions, report them

```
/* bpf_prog_load(): keep a copy */
if (IS_ENABLED(CONFIG_CHECKPOINT_RESTORE) &&
    type == BPF_PROG_TYPE_CGROUP_DEVICE &&
    prog->len ≤ BPF_MAXINSNS)
    prog->aux->orig_insns = kvmemdup(prog->insns,
...);

/* BPF_OBJ_GET_INFO_BY_FD: report it */
struct bpf_prog_info {
    ...
    __u32 orig_prog_len;
    __aligned_u64 orig_prog_insns;
};
```

## **CGROUP\_DEVICE, up to 4096 insns**

Covers every program a VE can load; at most 32 KB each.

## **Only with C/R enabled**

No cost without CONFIG\_CHECKPOINT\_RESTORE.

## **No bpf\_dump\_raw\_ok()**

The original program holds no kernel addresses.

## **Host only**

Still protected by bpf\_capable().



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



# Dump and restore

## Dump

1. Find programs through fds (anon\_inode:bpf-prog) and through BPF\_PROG\_QUERY on each cgroup v2 dir; the container root is skipped.
2. BPF\_OBJ\_GET\_INFO\_BY\_FD: save the original insns once per prog\_id.
3. Record fd entries and per-cgroup attachments with their attach\_flags.

## Restore

1. Load each program once with BPF\_PROG\_LOAD; the destination kernel re-verifies and re-JITs it.
2. Re-attach it to each cgroup dir once the dir exists, with the saved flags.
3. Reinstall fds as dups of the same loaded program.

Only CGROUP\_DEVICE is supported: any other program type fails the dump.



## DISCUSSION

# Problems

## Instructions are not portable

- **Constant blinding:** constants in xlated
- **Helper calls:** kernel address offsets in xlated
- **Context access:** kernel struct offsets in xlated
- **Spectre hardening:** masking insns in xlated
- **CO-RE:** relocated before load, for the current kernel
- **kfuncs, ksyms:** BTF IDs of one kernel build
- **kprobes:** the target must exist on the destination
- **Maps:** stale map fds; patch in the restored ones

## Around the program

- **bpffs:** programs pinned or linked there
- **bpffs itself:** restore the mount with delegate options
- **Tokens:** applications know nothing about them
- **Memory:** extra copy of the original insns
- **Program BTF:** dump and reload func and line info
- **Links:** owners and attach order
- **Object ids:** prog and map ids change on restore

For device programs in a VE the original copy already solves blinding; the rest is the general case.



LINUX  
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026





# Links

## KERNEL (VZKERNEL, VZ10)

bpf: report original insns of  
cgroup device programs, plus  
selftests

[openvz/vzkernel@b19c495a](mailto:openvz/vzkernel@b19c495a)

Pavel Tikhomirov · Virtuozzo · ptikhomirov@virtuozzo.com

## CRIU (OPENVZ)

bpfprog image, dump/restore,  
cgroup attachments, zdtm, crit

[openvz/criu.ovz@5f4c5bb2,  
d14097e2](mailto:openvz/criu.ovz@5f4c5bb2,d14097e2)

## THIS TALK

LPC 2026, Containers and  
Checkpoint/Restore MC

[lpc.events/event/20/contributions/  
2606](https://lpc.events/event/20/contributions/2606)



LINUX  
PLUMBERS  
CONFERENCE

PRAGUE, CZECHIA  
OCT. 5-7, 2026