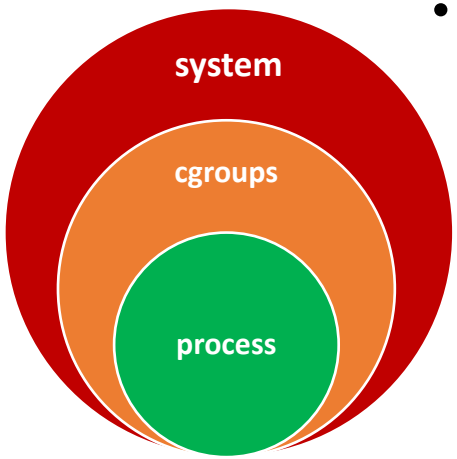
On access control model of Linux native performance monitoring

Motivation

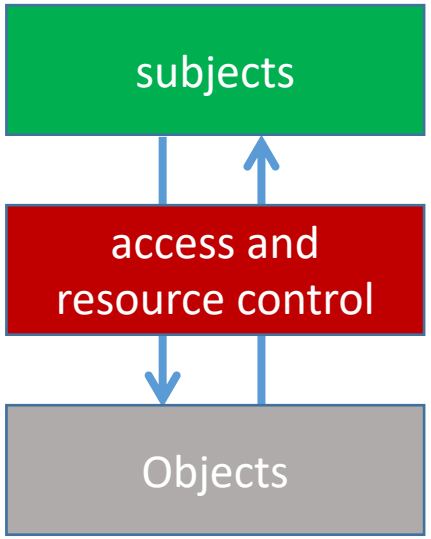
- socialize Perf access control management to broader community
- promote the management to security sensitive production environments
- discover demand on extensions to the existing Perf access control model

Model overview

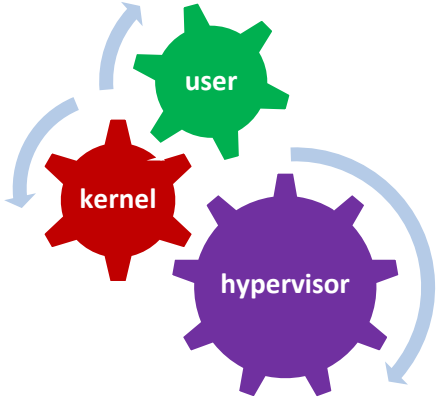
- **Subjects: processes**
  - superuser root
  - privileged user groups
  - unprivileged users
- **Objects: telemetry data**
  - tracepoints, OS events
  - CPU
  - Uncore
  - Other HW
- ***Scope***
  - process
  - cgroups
  - system



- **Access control:**
  - LSM hooks for MAC (e.g. SELinux)
  - Linux capabilities (DAC)
  - perf_event_paranoid sysctl
- **Resource control:**
  - CPU time: sample rate & throttling
  - Memory: perf_events_mlock_kb sysctl
  - File descriptors: ulimit -n (RLIMIT_NOFILE)



- ***Level***
  - user mode
  - kernel
  - hypervisor



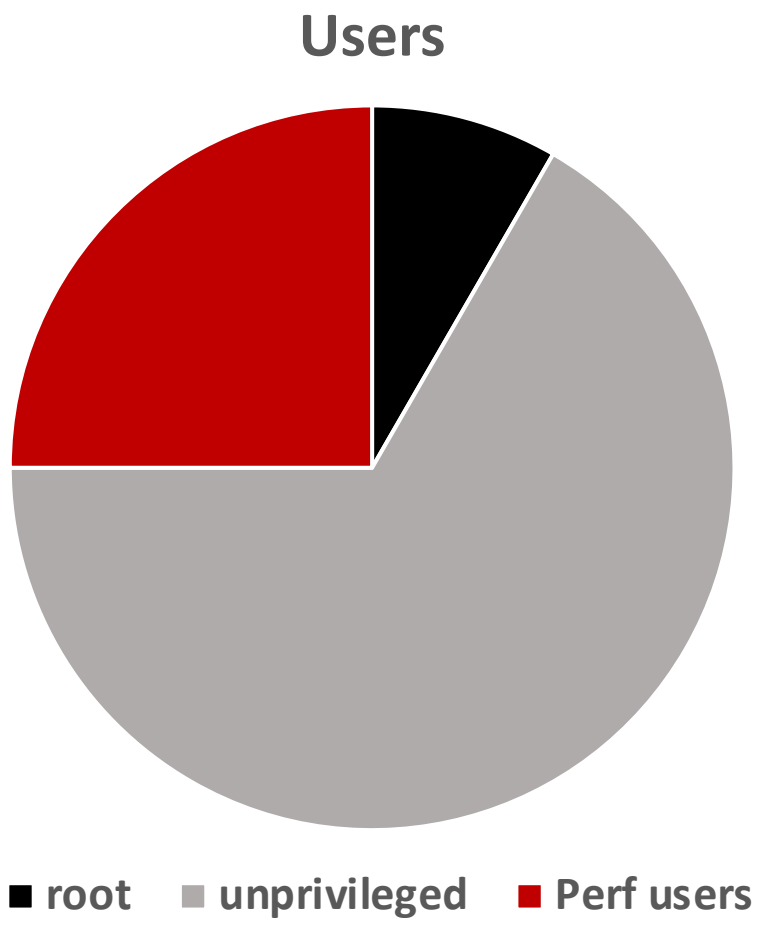
Subjects

- **root, superuser:**
  - euid = 0 and/or CAP_SYS_ADMIN
- **unprivileged users:**
  - perf_event_paranoid sysctl
- **Perf privileged user group:**

  -rwxr-x--- 2 root perf_users 11M Oct 19 15:12 perf

  # getcap perf

  perf = cap_perfmon,…=ep



Telemetry, scope, level

Objects: SW, HW telemetry data

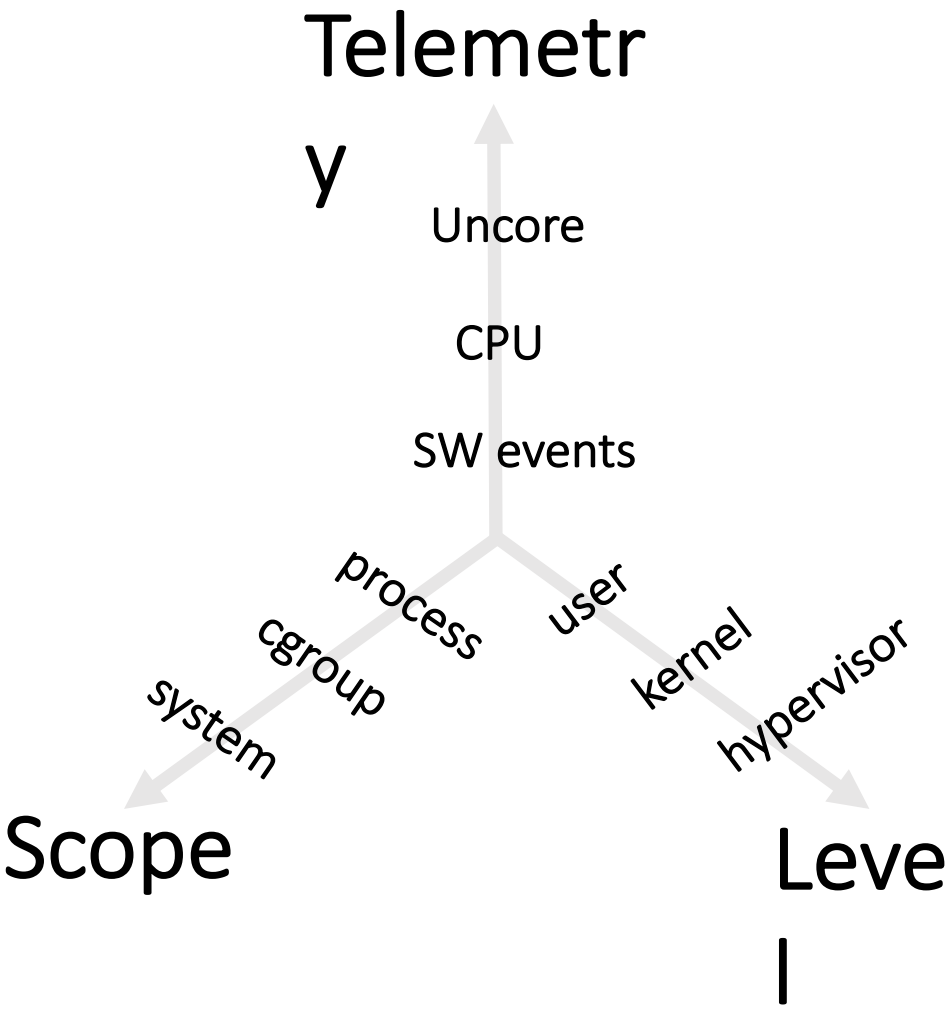
- tracepoints, OS events, eBPF
- CPUs events and related HW
- Uncore events (LLC, Interconnect, DRAM)
- Other (e.g. FPGA)

Scope:

- process
- cgroups
- system wide

Level:

- user
- kernel
- hypervisor



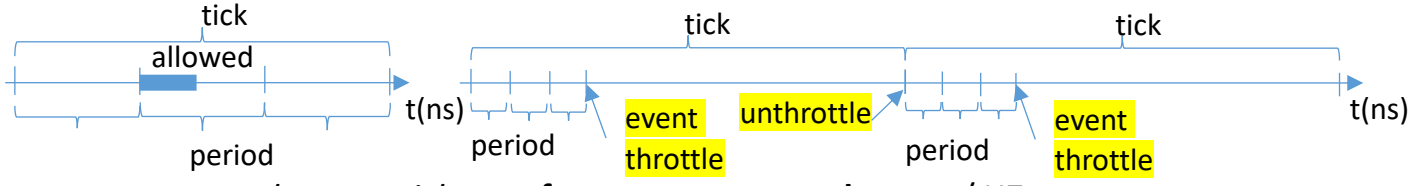
perf: interrupt took too long (3000 > 2000), lowering kernel.perf_event_max_sample_rate to 10000

Resource control

period = NSEC_PER_SEC / **perf_event_max_sample_rate,**
allowed = period * **perf_cpu_time_max_percent** / 100,
allowed_per_tick = (TICK_NSEC / 100) * **perf_cpu_time_max_percent,**
avg_len (#128) >= allowed => avg_len += 25%, avg_len < allowed_per_tick ?
max_samples_per_tick = allowed_per_tick / avg_len : *max_samples_per_tick* = 1,
allowed = avg_len, **perf_event_max_sample_rate** = *max_samples_per_tick* * HZ

CPU time: sample rate & throttling

- **perf_event_max_sample_rate** (Hz)
- **perf_cpu_time_max_percent** (%)



max_samples_per_tick = **perf_event_max_sample_rate** / HZ

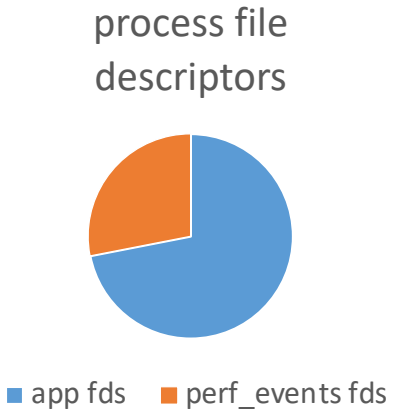
Memory:

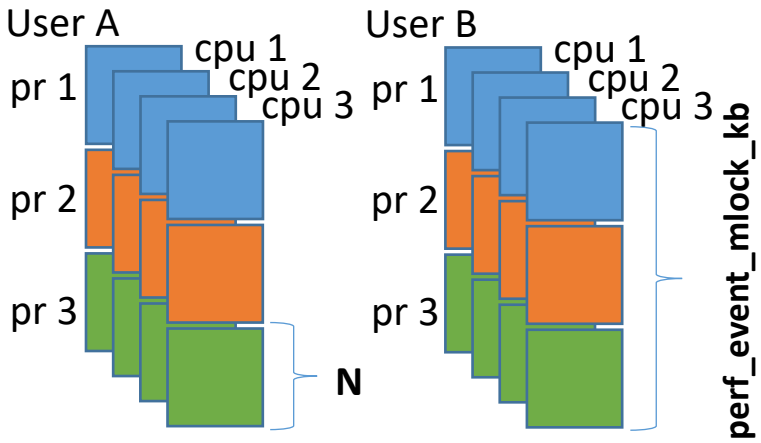
- **perf_event_mlock_kb** (KiB)
- perf record --mmap-pages **N**
- CAP_IPC_LOCK

Permission error mapping pages.
Consider increasing /proc/sys/kernel/perf_event_mlock_kb,
or try again with a smaller value of -m/--mmap_pages.
(current value: 4294967295,0)

File descriptors: **ulimit -n**

- /etc/security/limits.conf
- #events x #cpus





Too many events are opened.
Probably the maximum number of open file descriptors has been reached.
Hint: Try again after reducing the number of events.
Hint: Try increasing the limit with 'ulimit -n <limit>'

Access control

LSM hooks for MAC (v5.5+):

- open, read, write, tracepoint, kernel, cpu

Linux capabilities (DAC):

- CAP_PERFMON (v5.8+)
- CAP_SYS_ADMIN (v5.7-)
- CAP_SYS_PTRACE
- CAP_SYSLOG
- CAP_SYS_RAWIO

Sysctl:

- perf_event_paranoid

```
Access to performance monitoring and observability operations is limited.
Enforced MAC policy settings (SELinux) can limit access to performance
monitoring and observability operations. Inspect system audit records for
more perf_event access control information and adjusting the policy.
Consider adjusting /proc/sys/kernel/perf_event_paranoid setting to open
access to performance monitoring and observability operations for processes
without CAP_PERFMON, CAP_SYS_PTRACE or CAP_SYS_ADMIN Linux capability.
More information can be found at 'Perf events and tool security' document:
https://www.kernel.org/doc/html/latest/admin-guide/perf-security.html
perf_event_paranoid setting is -1:
   -1: Allow use of (almost) all events by all users
       Ignore mlock limit after perf_event_mlock_kb without CAP_IPC_LOCK
>= 0: Disallow raw and ftrace function tracepoint access
>= 1: Disallow CPU event access
>= 2: Disallow kernel profiling
To make the adjusted perf_event_paranoid setting permanent preserve it
in /etc/sysctl.conf (e.g. kernel.perf_event_paranoid = <setting>)
```

https://www.kernel.org/doc/html/latest/admin-guide/perf-security.html

Unprivileged users: perf_event_paranoid

-1: no scope and level restrictions. No memory limit but open files limit is applied.

>= 0: no scope and level restrictions but raw and ftrace tracepoints are not accessible.
Memory and open files limits are applied.

>= 1: own process monitoring only. No level restrictions. Memory and open files limits are applied.

>= 2: own process monitoring only. User mode level only. Memory and open files limits are applied.

perf_event_paranoid	process			cgroups			system		
	user	kernel	VM, HV	user	kernel	VM, HV	user	kernel	VM, HV
raw and ftracepoints	=-1	=-1	=-1	=-1	=-1	=-1	=-1	=-1	=-1
CPU events	<=2	<=1	<=1	<=0	<=0	<=0	<=0	<=0	<=0
Uncore events	<=0	<=0	<=0	<=0	<=0	<=0	<=0	<=0	<=0
Other HW	<=0	<=0	<=0	<=0	<=0	<=0	<=0	<=0	<=0

production: accessible likely accessible unlikely accessible not accessible

Privileged user groups: CAP_PERFMON

-rwxr-x--- 2 root **perf_users** 11M Aug 20 00:00 perf

setcap "cap_perfmon,cap_sys_ptrace,cap_syslog,cap_sys_rawio=ep" perf

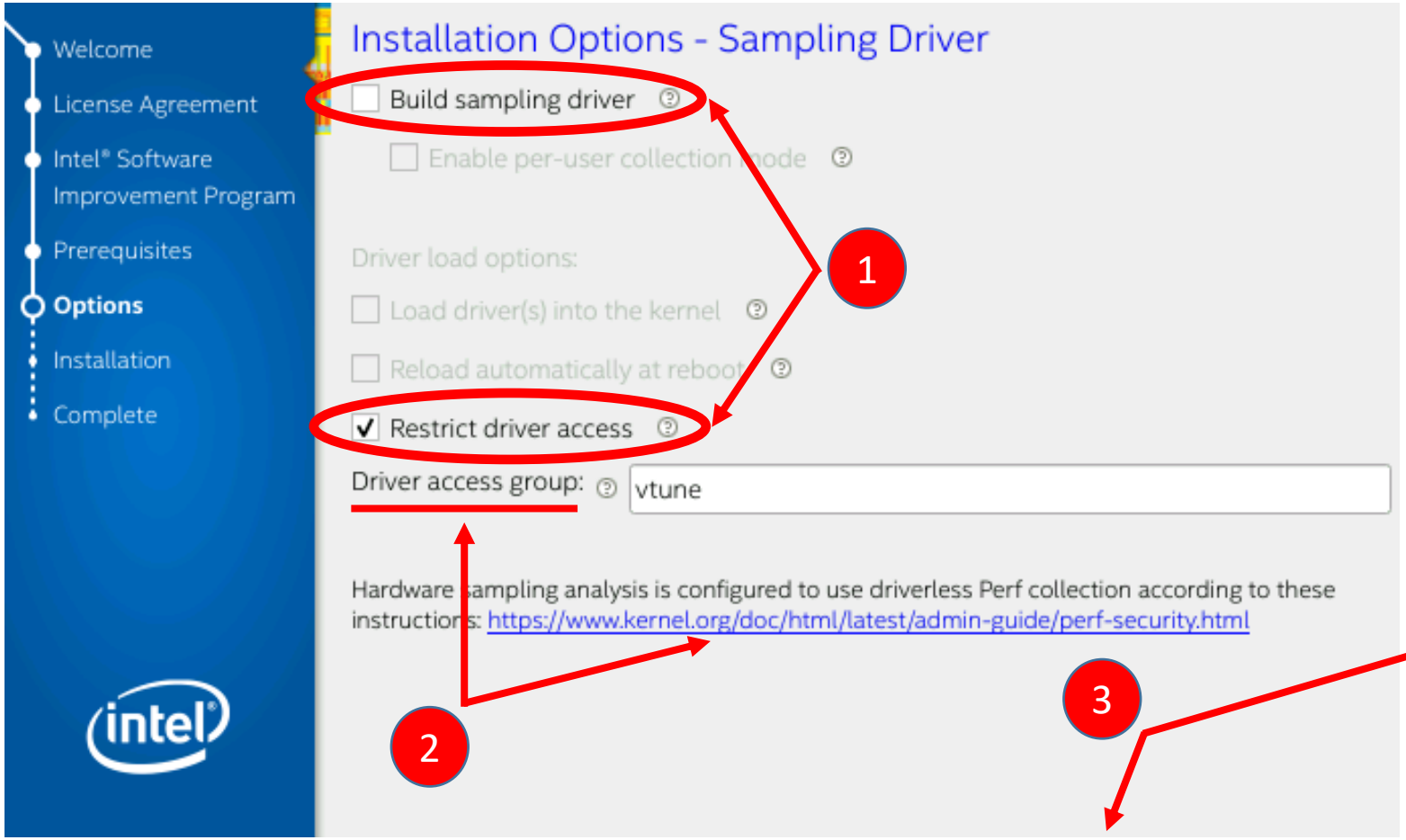
getcap perf

perf = cap_sys_rawio,cap_sys_ptrace,cap_syslog,cap_perfmon+ep

CAP_PERFMON, CAP_SYS_PTRACE,…	**process**			**cgroups**			**system**		
	user	kernel	VM, HV	user	kernel	VM, HV	user	kernel	VM, HV
raw and ftracepoints	Y,Y,…	Y,Y,…	Y,Y,…	Y,Y,…	Y,Y,…	Y,Y,…	Y,Y,…	Y,Y,…	Y,Y,…
CPU events	Y,Y,…	Y,Y,…	Y,Y,…	Y,Y,…	Y,Y,…	Y,Y,…	Y,Y,…	Y,Y,…	Y,Y,…
Uncore events	Y,Y,…	Y,Y,…	Y,Y,…	Y,Y,…	Y,Y,…	Y,Y,…	Y,Y,…	Y,Y,…	Y,Y,…
Other HW	Y,Y,…	Y,Y,…	Y,Y,…	Y,Y,…	Y,Y,…	Y,Y,…	Y,Y,…	Y,Y,…	Y,Y,…

production: accessible likely accessible unlikely accessible not accessible

Perf privileged user groups @ Intel VTune



- **full access to Linux Perf monitoring capabilities**
- **no perf_event_paranoid limitations**
- **secure, flexible, manageable access control**
- **follows traditional Linux security model**
- **commercial grade, documented performance analysis capabilities**

https://software.intel.com/content/www/us/en/develop/documentation/vtune-cookbook/top/configuration-recipes/profiling-hardware-without-sampling-drivers.html

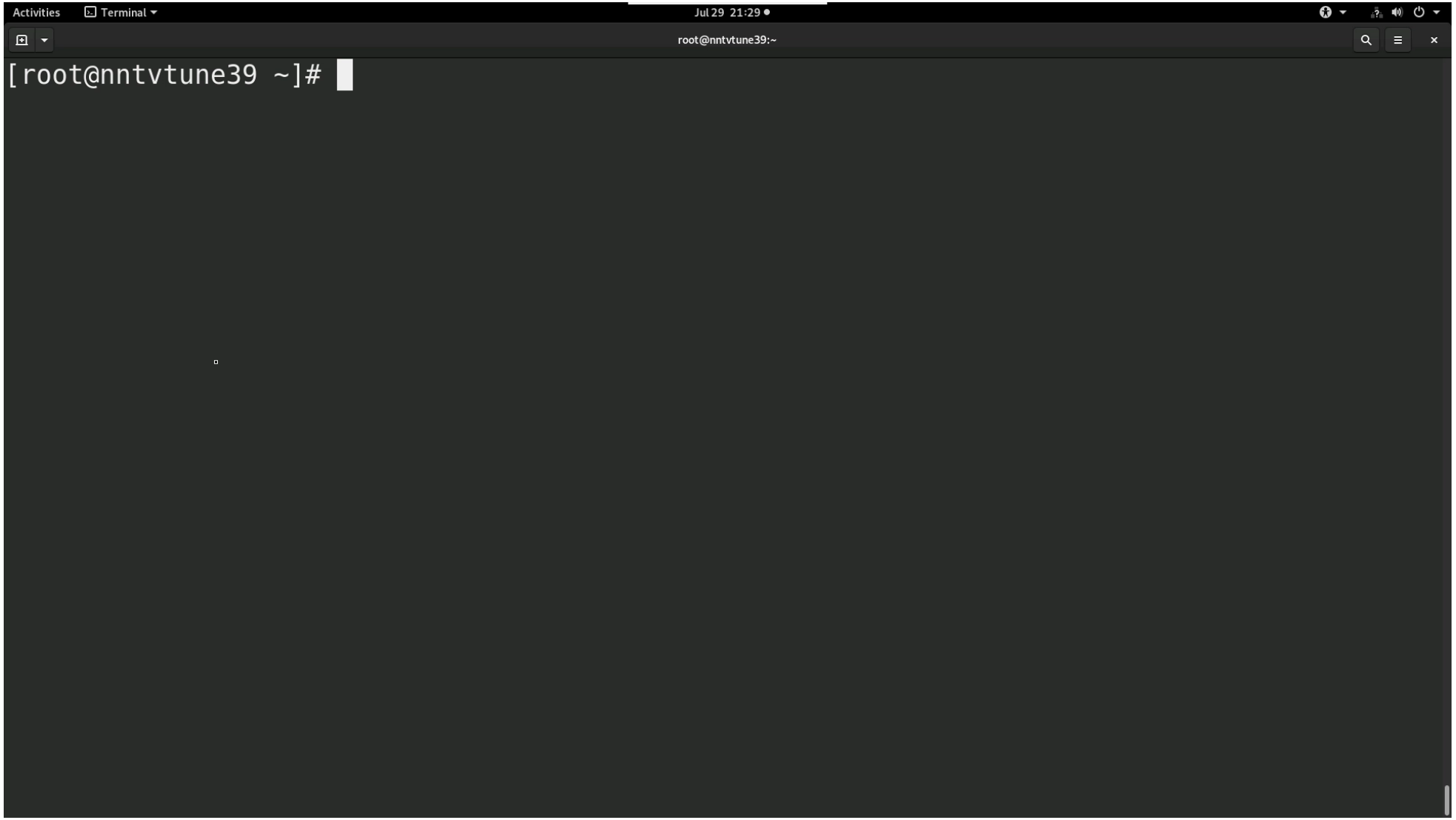
MAC to Linux Perf

Access to performance monitoring and observability operations is limited. Enforced MAC policy settings (SELinux) can limit access to performance monitoring and observability operations. Inspect system audit records for more perf_event access control information and adjusting the policy.

```
# journalctl --reverse --no-pager | grep perf_event
setroubleshoot[1318099]: SELinux is preventing perf from open access on the perf_event labeled …
audit[1318098]: AVC avc:  denied  { open } for  pid=1318098 comm="perf"  …
# ausearch -c 'perf' --raw | audit2allow -M my-perf.te
# checkmodule -M -m -o my-perf.mod my-perf.te
# semodule_package -o my-perf.pp -m my-perf.mod
# semodule -X 300 -i my-perf.pp
# semodule -d my-perf
# semodule -e my-perf
```

<kernel_source>/tools/perf/Documentation/security.txt

Demo


Activities
Terminal
Jul 29 21:29
root@nntvtune39:~
[root@nntvtune39 ~]#

Q/A

Intel VTune users:
https://community.intel.com/t5/Analyzers-Intel-VTune-Profiler/bd-p/analyzers

Linux Perf users: linux-perf-users@vger.kernel.org
Linux Perf development: linux-kernel@vger.kernel.org

Contact: Alexei Budankov alexey.budankov@linux.intel.com

Acknowledgements:
Stephen Smalley, Casey Schaufler, Serge Hallyn, James Morris,
Lionel Landwerlin, Alexei Starovoitov, Arnaldo Carvalho De Melo, Jiri Olsa,
Andi Kleen and other community folks …

Backup

my-perf.te

```
module my-perf 1.0;
require {
	type unconfined_t;
	class perf_event { cpu kernel open read tracepoint write };
}
#============== unconfined_t ===============
allow unconfined_t self:perf_event { cpu kernel open read tracepoint write };
```