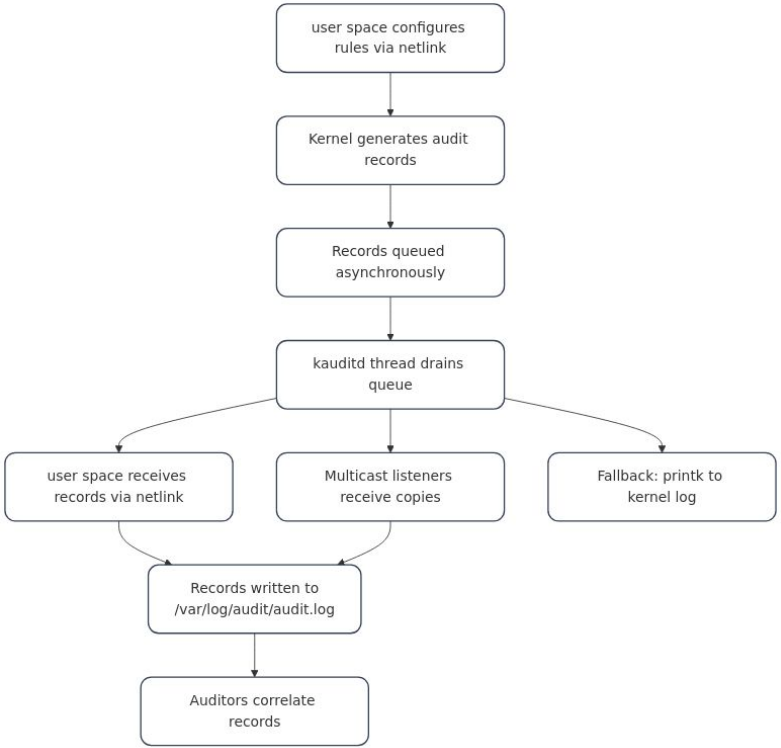
CLOUDFLARE

# BPF LSMs + Audit

**Frederick Lawler**
Linux Team
Cloudflare

**Linux Security Summit**
Minneapolis, MN, US
May 2026

# Audit: All about observability



CLOUDFLARE

# Audit user space integration

### Auditd

- Binds daemon process to audit subsystem
- Consumes Audit messages via Netlink API
- Integrates with plugins to direct output to logs on file system

### Auditctl

- Configures audit subsystem
- Configures filter rules

### Ausearch/aureport

- Convenient tools to search audit logs

CLOUDFLARE

# ausearch -ui 1000

```
----

time->Mon May 11 13:12:04 2026

type=PROCTITLE msg=audit(1778523124.031:688491):
proctitle=2F736E61702F736E6170642F32363836352F7573722F6C69622F736E6170642F736E61702D636F6E66696E65002D
2D6261736500636F7265323200736E61702E63616E6F6E6963616C2D6C697665706174636 82E63616E6F6E6963616C2D6C6976
657061746368002F7573722F6C69622F736E6170642F736E61702D65786563

type=UNKNOWN[1420] msg=audit(1778523124.031:688491):
subj_apparmor=/snap/snapd/26865/usr/lib/snapd/snap-confine

type=SYSCALL msg=audit(1778523124.031:688491): arch=c000003e syscall=321 success=yes exit=9 a0=5
a1=7fff06a78370 a2=80 a3=f items=0 ppid=7817 pid=2373820 auid=1000 uid=1000 gid=1000 euid=1000
suid=1000 fsuid=1000 egid=1000 sgid=1000 fsgid=1000 tty=(none) ses=3 comm="snap-confine"
exe="/snap/snapd/26865/usr/lib/snapd/snap-confine" subj=? key=(null)

type=BPF msg=audit(1778523124.031:688491): prog-id=38622 op=LOAD

...
```

CLOUDFLARE

# A BPF LSM in the wild

```
SEC("lsm/userns_create")
int BPF_PROG(handle_userns_create, struct cred *cred, int ret)
{
    if (ret)
        return ret;

    struct task_struct *task = bpf_get_current_task_btf();

    if (task_in_allow_list(task))
        return 0;

    // What was denied?

    return -EPERM;
}
```

CLOUDFLARE

# Assuming a syscall exit rule exists

```
----

time->Tue May 12 13:43:13 2026

type=PROCTITLE msg=audit(1778611393.272:28):
proctitle=756E7368617265002D7255

type=SYSCALL msg=audit(1778611393.272:28): arch=c000003e syscall=272
success=no exit=-1 a0=10000000 a1=7ffe829fa390 a2=0 a3=8 items=0 ppid=518
pid=538 auid=4294967295 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0
fsgid=0 tty=pts0 ses=4294967295 comm="unshare" exe="/usr/bin/unshare"
key="unshare"
```

CLOUDFLARE

# BPF does offer...

## BPF Ring buffer (fast, custom consumer)

```
struct {
   __uint(type, BPF_MAP_TYPE_RINGBUF);
   __uint(max_entries, 256 * 1024);
} ringbuf SEC(".maps");

SEC("lsm/bprm_check_security")
int BPF_PROG(handle_exec, struct linux_binprm *bprm, int ret)
{
   e = bpf_ringbuf_reserve(&ringbuf, sizeof(*e), 0);
   if (!e)
       return 0;

   __u64 pid_tgid = bpf_get_current_pid_tgid();
   __u64 uid_gid  = bpf_get_current_uid_gid();
   e->pid = pid_tgid >> 32;
   e->uid = uid_gid & 0xFFFFFFFF;

   bpf_get_current_comm(&e->comm, sizeof(e->comm));
   const char *filename_ptr = BPF_CORE_READ(bprm, filename);
   bpf_probe_read_kernel_str(e->filename, sizeof(e->filename),
                             filename_ptr);

   bpf_ringbuf_submit(e, 0);
   return 0;
}
```

## bpf_printk() (slower, tracefs)

```
SEC("lsm/bprm_check_security")
int BPF_PROG(handle_exec, struct linux_binprm *bprm, int ret)
{
   char filename[256];
   char comm[16];

   __u64 pid_tgid = bpf_get_current_pid_tgid();
   __u64 uid_gid  = bpf_get_current_uid_gid();
   __u32 pid = pid_tgid >> 32;
   __u32 uid = uid_gid & 0xFFFFFFFF;

   bpf_get_current_comm(&comm, sizeof(comm));

   const char *filename_ptr = BPF_CORE_READ(bprm, filename);
   bpf_probe_read_kernel_str(filename, sizeof(filename),
                             filename_ptr);

   bpf_printk("audit: pid=%u uid=%u comm=%s file=%s", pid, uid,
              comm, filename);

   return 0;
}
```

CLOUDFLARE

# The problem with BPF LSM logs today

- Bespoke userspace tooling
- Fragmented context
- Filtering not included
- No coordination among N BPF-LSM policies
- Log rotation, policy updates & restarts, a mess
- Where do auditors look?

CLOUDFLARE

# On adding security_create_user_ns()

For observably this is a terrible LSM interface because there is no pair with user namespace destruction, nor is their any ability for the LSM to allocate any state to track the user namespace.

**As there is no patch actually calling audit or anything else observably does not appear to be a driving factor of this new interface.**

https://lore.kernel.org/all/8735dux60p.fsf@email.froward.int.ebiederm.org/

CLOUDFLARE

# Proposing BPF + Audit

```
SEC("lsm/bprm_check_security")
int BPF_PROG(handle_exec, struct linux_binprm *bprm, int ret)
{
  struct bpf_audit_context *ac;
  if (ret != 0)
    return ret;

  ac = bpf_audit_log_start();
  if (!ac)
    return -12;

  bpf_audit_log_end(ac);

  return 0;
}
```

https://lore.kernel.org/all/20260311-bpf-auditd-send-message-v1-0-10a62db5c92f@cloudflare.com/

CLOUDFLARE

# ausearch -m 1427

```
----

time->Tue May 12 10:21:22 2026

type=PROCTITLE msg=audit(1778599282.491:57): proctitle="ls"

type=PATH msg=audit(1778599282.491:57): item=1 name="/lib64/ld-linux-x86-64.so.2" inode=32770355 dev=00:15
mode=0100755 ouid=0 ogid=0 rdev=00:00 nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=0

type=PATH msg=audit(1778599282.491:57): item=0 name="/usr/bin/ls" inode=32792017 dev=00:15 mode=0100755 ouid=0
ogid=0 rdev=00:00 nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=0

type=CWD msg=audit(1778599282.491:57): cwd="/home/fred/Presentations/bpf-audit"

type=EXECVE msg=audit(1778599282.491:57): argc=1 a0="ls"

type=SYSCALL msg=audit(1778599282.491:57): arch=c000003e syscall=59 success=yes exit=0 a0=56203525dc10
a1=562035257530 a2=562035360250 a3=562035257530 items=2 ppid=404 pid=428 auid=4294967295 uid=0 gid=0 euid=0
suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts0 ses=4294967295 comm="ls" exe="/usr/bin/ls" key="exec"

type=UNKNOWN[1427] msg=audit(1778599282.491:57): prog-id=13 pid=428 comm="bash"
```

```
AUDIT_BPF_LSM_ACCESS 1427 /* LSM BPF MAC events */
```

# Discussion topics

CLOUDFLARE

# Exported functions

- Wraps & demultiplex most audit_log_lsm_data() functionality
- bpf_audit_log_*() functions limited to 1 invocation each
- Functions such as bpf_audit_log_path() & bpf_audit_log_file() treated as same function call

Implementation Link

**Unexported**

- LSM_AUDIT_DATA_ANONINODE
- LSM_AUDIT_DATA_IBENDPORT
- LSM_AUDIT_DATA_IBPKEY
- LSM_AUDIT_DATA_IPC
- LSM_AUDIT_DATA_KMOD
- LSM_AUDIT_DATA_LOCKDOWN
- LSM_AUDIT_DATA_NLMSGTYPE
- LSM_AUIDT_DATA_KEY
- audit_log_format()

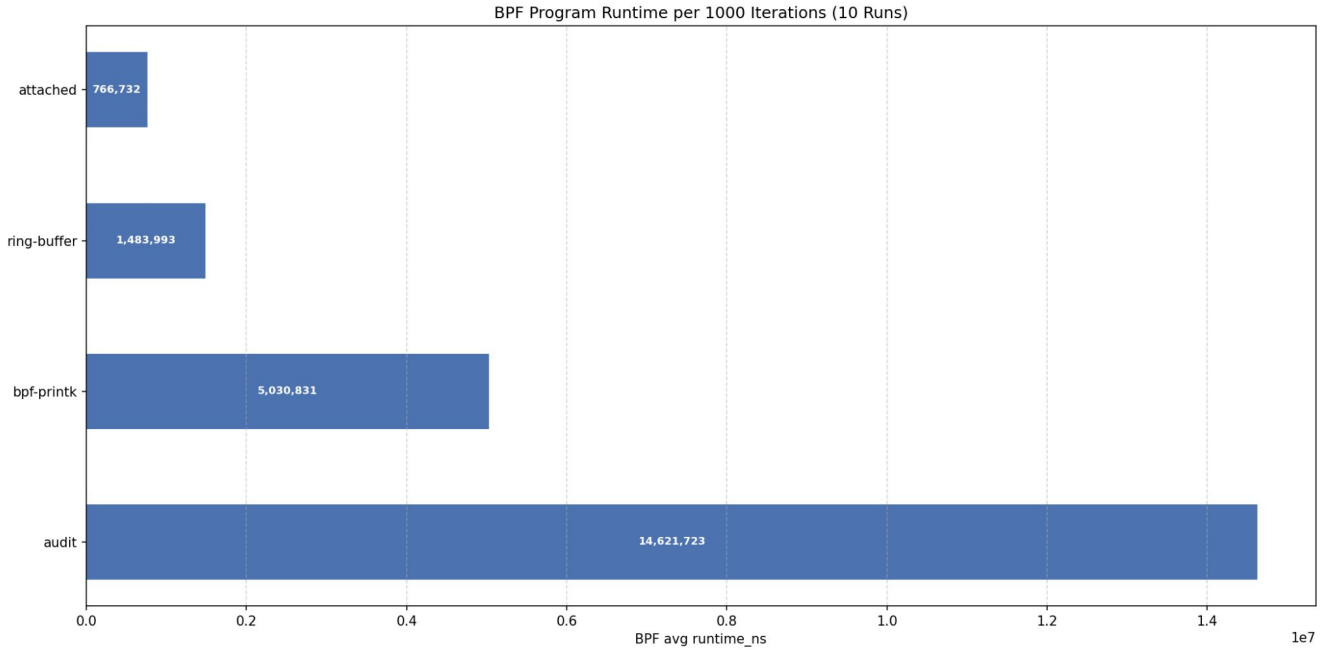
CLOUDFLARE

# bpf_audit_log_start()

- Detects if BPF program is executed in sleepable context & allocates struct bpf_audit_context accordingly
- audit_context() is called on audit_log_start()
  - Should users decide if they want this?
- BPF bpf_audit_log_start() error handling
  - Assume return current allow/deny code?
  - New API to retrieve BPF audit error code for separate handling?

CLOUDFLARE

# Crazy synthetic benchmark




Audit: min=6,619,263; max=26,844,367 runtime_ns

CLOUDFLARE

# Dive in: a small sample

```
3)               |  bpf_audit_log_start() {
3)               |    audit_log_start() {
3)               |      audit_filter() { …
3)   1.050 us    |      }
3)               |      auditd_test_task() { …
3)   1.031 us    |      }
3)               |      kmem_cache_alloc_noprof() { …
3)   0.887 us    |      }
3)   0.172 us    |      __raw_spin_lock_init();
3)               |      __alloc_skb() {
3)   0.160 us    |        __local_bh_disable_ip();
3)   0.267 us    |        __local_bh_enable_ip();
3)               |        kmem_cache_alloc_node_noprof() { …
3)   1.109 us    |        }
3)               |        kmalloc_reserve() {
3)   0.165 us    |          kmalloc_size_roundup();
3)               |          __kmalloc_node_track_caller_noprof() { …
3)   0.988 us    |          }
3)   1.801 us    |        }
3)   4.460 us    |      }
3)               |      skb_queue_tail() { …
3)   1.664 us    |      }
3)               |      __nlmsg_put() { …
3)   0.432 us    |      }
3)   0.139 us    |      auditsc_get_stamp();
3)               |      audit_log_format() { …
3)   0.899 us    |      }
3) + 12.451 us   |    }
3)               |    audit_log_format() { …
3)   0.783 us    |    }
3)               |    audit_log_format() { …
3)   0.764 us    |    }
3)               |    audit_log_untrustedstring() { …
3)   0.978 us    |    }
3) + 21.538 us   |  } <~~~~~~~ BPF stuff included
```

```
3)               |  bpf_audit_log_end() {
3)               |    audit_log_end() {
3)               |      audit_log_end.part.0() { <~~~~~ Little bit of looping
3)               |        skb_dequeue() { …
3)   1.464 us    |        }
3)               |        skb_queue_tail() { …
3)   1.448 us    |        }
3)               |        skb_dequeue() { …
3)   1.413 us    |        }
3)               |        __wake_up() {
…
3)               |          __wake_up_common() {
3)               |            autoremove_wake_function() {
3)               |              default_wake_function() {
3)               |                try_to_wake_up() {
…
3)   0.129 us    |                  kthread_is_per_cpu();
3)               |                  ttwu_queue_wakelist() {
3)               |                    __smp_call_single_queue() {
3)   0.117 us    |                      call_function_single_prep_ipi();
3)               |                      native_send_call_func_single_ipi() {
3)   1.318 us    |                        x2apic_send_IPI();
3)   1.574 us    |                      }
3)   2.251 us    |                    }
3)   2.608 us    |                  }
…
3)   5.613 us    |                }
3)   5.883 us    |              }
3)   6.180 us    |            }
3)   6.546 us    |          }
…
3)   9.391 us    |        }
…
3) + 16.841 us   |      }
3) + 17.141 us   |    }
3) + 17.516 us   |  }
```

CLOUDFLARE

# Spitballing

- Should this API attempt to minimize allocations & kauditd wakeup by moving to a dedicated workqueue for the BPF implementation?
- IMA exposes an attested event log, bpf_printk() hooks into the tracing subsystem. Could we expose /sys/kernel/security/audit/log?
  (yes, this does imply years of work needed)

CLOUDFLARE

# Current patch state

- Selftests are being re-worked to address CI errors
  - Likely to extract Lockdown's implementation to maximize reuse
- Proper BPF selftests benchmarking to be added
- Minor fixup recommendations
- Rework & benchmarking is needed to address performance concerns (looking for feedback/ideas)

# Thank you

fred@cloudflare.com