Google

BPF + Security @ Google

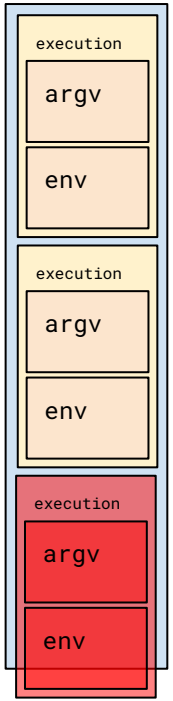
Brendan Jackman (jackmanb@google.com)
KP Singh (kpsingh@google.com)
Linux Plumbers Conference 2021

Agenda

- Atomics and Promises
- Chunking data + BPF ring buffer
- What next for BPF security auditing?
- What's missing for implementing enforcement policies?

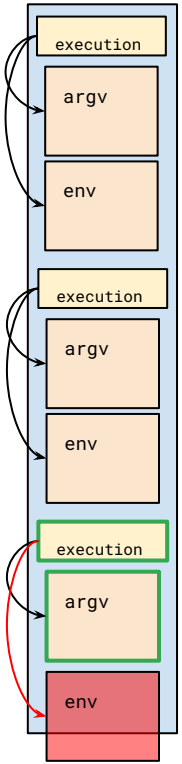
Why did we add atomics to BPF?

Promises



Some of our events are pretty big!

Promises



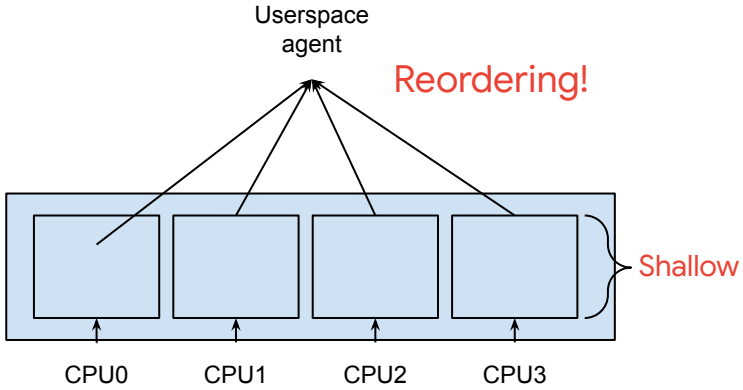
It’s pretty useful to break things up into smaller chunks.

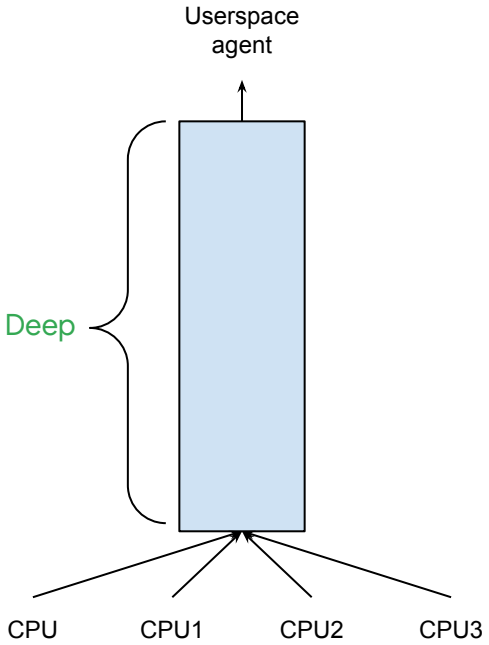
We use unique IDs to connect the chunks. We call this connection a “promise” - like in async frameworks.

That’s why we needed atomics.

Why do we like the BPF ringbuf?

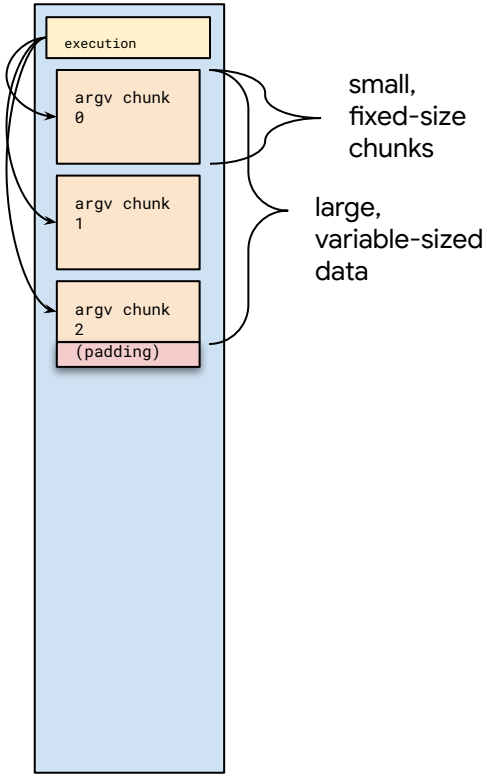
Ring buffers: perf buffer vs BPF ringbuf





Ring buffer chunking trick

Ring buffers: chunking



- Verifier likes to know buffer sizes in advance
- But allocating max-possible size is bad
- Break down large data into fixed-size chunks

What's next for BPF security auditing?

BPF LSM for Auditing - Current State

We don't audit through BPF LSM as much as we'd like. Some examples:

Info	Why not LSM?	Current source	Problem
Process execution	N/A	BPF LSM	N/A
Mmap	Missing vma	Perf (not BPF)	Inflexible: missing data
Socket ops	Missing e.g. port	Tangle of fexit hooks	Maintenance
Module load	Missing name	Tracepoints (BPF)	Inflexible: missing data

Experience: Auditing with BPF

Currently there is no clean and flexible surface to attach to

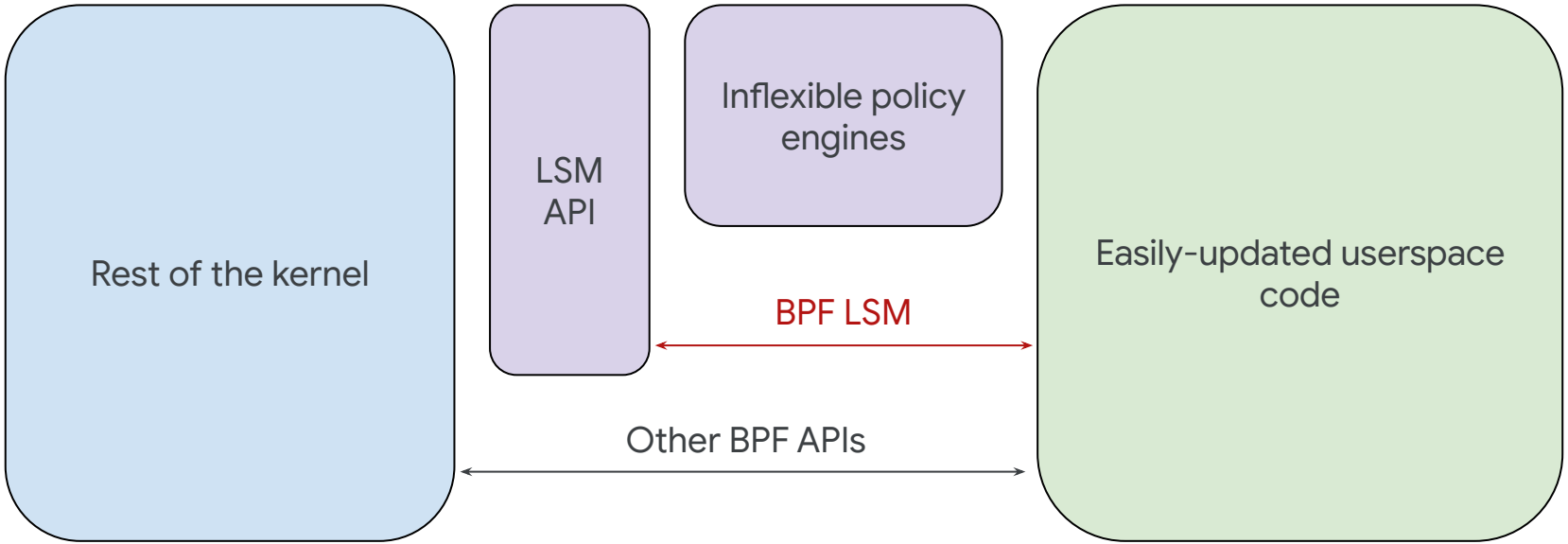
LSM provides a bespoke surface for *enforcement*.

That surface **captures value** created by enforcement experts

Do we want a bespoke surface for *auditing*?

To **capture value** created by auditing experts

Big picture: BPF LSM



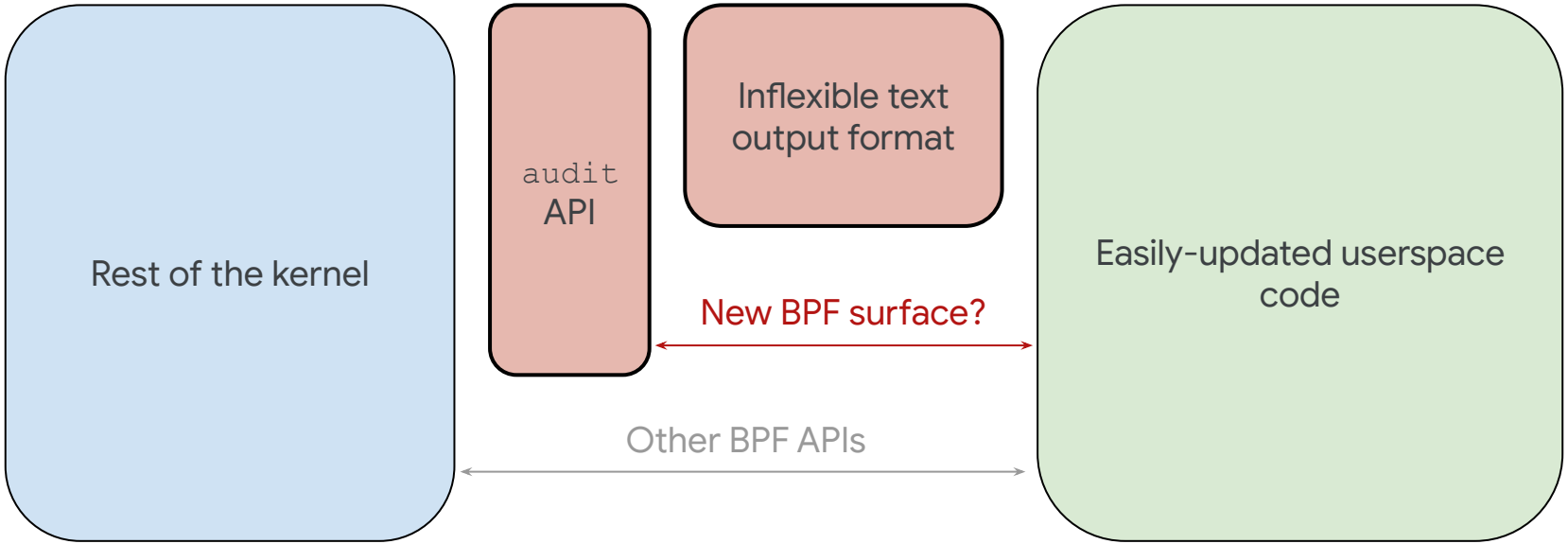
Add new LSM hooks

- Add new LSM hooks at these places
- These new LSM hooks are for bookkeeping only:
  - There are other such hooks:
    - blob/state management
    - Introduced for specific LSMs needs

Pro: Easy to implement

Con: Currently not tied to an existing MAC policy (but they could be)

Big picture: BPF audit



Expose audit events to BPF

- Work needed to get BPF attachment points (currently all static inline)
- Existing surfaces is exactly what is needed for the text output format. Would need to extend it significantly.

Pro: Existing surface
Con: Major overhaul of audit

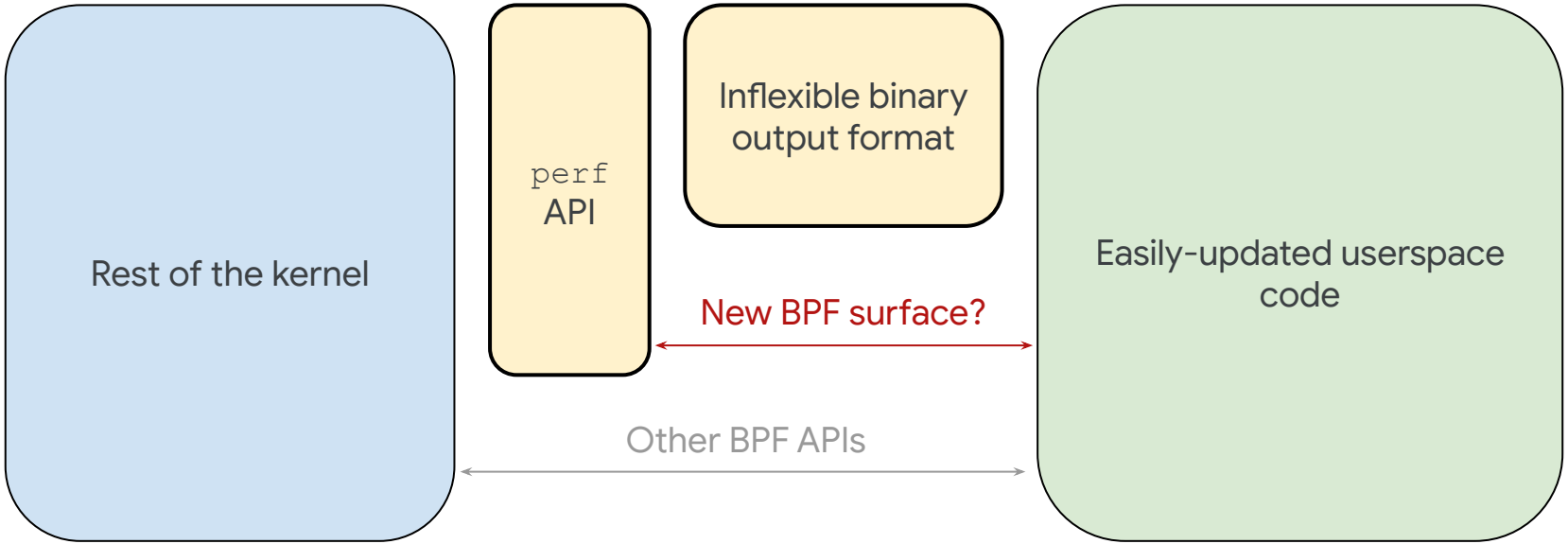
Expose perf events to BPF

- Existing API for:
  - mmap
  - changes to kernel text
  - namespaces
  - fork, exit, exec
  - bpf program load and unload
- FTrace CFLAGS are currently disabled on perf functions

Pro: Existing surface (with rich arguments)

Con: New events will need perf implementation + userspace changes

Big picture: BPF perf



What's missing for advanced enforcement?

Persistent security tags

- Required to persist security state across reboots
- LSMs use security labels implemented using xattrs
- BPF LSM cannot read or write xattrs
- Helpers needed!
  - `bpf_get_xattr`
  - `bpf_set_xattr`