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!
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

Reordering!
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:

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

- Rest of the kernel
- LSM API
- Inflexible policy engines
- Easily-updated userspace code
- Other BPF APIs
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

Rest of the kernel

Audit API

Inflexible text output format

New BPF surface?

Other BPF APIs

Easily-updated userspace code
Exposé les événements d'audit aux BPF

- Travail nécessaire pour obtenir des points d'attache BPF (actuellement tous statiques inline)
- Les surfaces existantes sont exactement ce qui est nécessaire pour le format de sortie de texte. Il faudrait l'élargir considérablement.

Pro: Les surfaces existantes
Con: Grand redimensionnement de l'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

- Rest of the kernel
- perf API
  - Inflexible binary output format
- Easily-updated userspace code
- Other BPF APIs

New BPF surface?
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