eBPF Notes

TJ Robinson

October 29, 2025

1 What is eBPF?

- eBPF (extended Berkeley Packet Filter)
- Outperforms *IPtables* based soliutions

1.1 Logs vs. Metrics vs. Observability

- Logs: Detailed, unstructured, aggregated data about individual events.
  - Useful for troubleshooting and forensic analysis.
  - Can be voluminous and harder to analyze at scale.
- Metrics: Aggregated, structured data for to monitor a program or system performance at a specific point in time.
  - Useful for identifying trends and triggering alerts.
  - Typically less detailed but more efficient to store and query.
- Observability: Combines logs, metrics, and traces to provide a comprehensive view of system behavior.
  - Capacity to ask arbitrary questions and recieve complex answers about a system's state.

1.2 Namespaces & Cgroups

Both are fundamental for containerization and resource management in Linux.

- Namespaces: Isolate system resources for processes (e.g., PID, network, mount).
  - inside a namespace, you experience the operating system like there were no other tasks running on the computer
- Cgroups: Control and limit resource usage (CPU, memory, I/O) for process groups.
  - gives you fine grain control over resource usage like CPU, disk I/O, network, and etc.

Tracepoints are static marks in the kernel code that can be used to inject code to inspect the kernel's execution.

2 BPF Program Types

2.1 Socket Filter Programs

- Attach to network sockets to filter packets.
- Commonly used for packet filtering and monitoring.
- You can not modify the packets, only accept, drop, forward, or observe them.

2.2 Kprobes and Uprobes

- Kprobes: Attach to kernel functions to trace and monitor kernel events.
- Uprobes: Attach to user-space functions to trace and monitor application events.

2.3 XDP *(eXpress Data Path)* Programs

- Attach to the earliest point in the network stack.
- Used for high-performance packet processing.
- Can modify or drop packets before they reach the kernel networking stack.