Some research on possible implementation approaches

Setting Up Monitoring for iSCSI

To monitor iSCSI kernel interactions:

1. **Identify Relevant Kernel Functions:** Determine which kernel functions handle iSCSI operations. This might include functions related to the SCSI subsystem and network stack.

2. **Write eBPF Programs:** Using `bcc` or `bpftrace`, write eBPF programs that attach to these kernel functions to trace events and collect metrics.GitHub+1BetterStack+1

3. **Export Metrics:** Utilize `ebpf_exporter` to export collected metrics to Prometheus, enabling visualization and alerting.srodi.com+2GitHub+2GitHub+2

4. **Visualize Data:** Use Grafana or Netdata to create dashboards that display the metrics, helping in monitoring and troubleshooting

bcc Python scripts: Flexible, easier for exporting custom metrics.

bpftrace scripts: Quick for prototyping, ideal for low-volume metrics.

CO-RE C eBPF + ebpf_exporter: Best for production, Prometheus-ready.

1. ebpf_exporter (Cilium)

- **Purpose:** Exports eBPF map data as Prometheus metrics.
- **Workflow:** Write & compile eBPF C programs, configure metric mapping in `ebpf_exporter.yaml`, run exporter.
- **Prometheus Integration:** Native `/metrics` endpoint.

LINKS : https://github.com/cloudflare/ebpf_exporter

---

2. BCC with Node Exporter (Textfile Collector)

- **Purpose:** Custom metric gathering via Python-based eBPF scripts.
- **Use Case:** Lightweight scripts for specific kernel events like iSCSI reads/writes.
- **Workflow:** Attach BCC to probes, write output to `.prom` files in Node Exporter's `textfile_collector` directory.
- **Prometheus Integration:** Indirect (via textfile collector).

---

3. bpftrace

- **Purpose:** Interactive tracing for debugging or short-term inspection.
- **Use Case:** Real-time kernel tracing and observation.
- **Workflow:** Use high-level scripts or one-liners to attach to iSCSI-related functions.
- **Prometheus Integration:** Not native; export requires custom bridging.
