Trident

Trident is the transport backbone shared by Kraken and standalone tooling. It wraps transports in a single `Conduit` abstraction so modules can hop between plain TCP, TLS, UDP, or DTLS without rewriting socket code.

At its core Trident exposes two conduit families:

- **Stream conduits** — connection-oriented stacks such as `tcp`, `tls over tcp`.
- **Datagram conduits** — message-oriented stacks such as `udp` or `dtls over udp`.

Every conduit reports its stack (e.g., `["tls","tcp"]`) and lifecycle (`Dial`, `Close`). This allows orchestrators to swap transports at runtime without the module noticing.

Building Stacks

Trident favors composition. A stack is expressed by nesting conduits:

1. Pick a base transport (`tcp` or `udp`).
2. Decorate it with higher layers (`tls`, `dtls`, logging, fuzzing, replay, etc.).
3. Hand the resulting conduit to the module, which only cares about the generic stream or datagram interface.

Because every layer reports the same metadata (start/end timestamps, local/remote addresses, protocol hints) you can log or replay sessions consistently across stacks.

Buffering & Metadata

- **Pooled buffers** keep allocations predictable. Receives hand back a buffer that callers release when done; sends can either borrow a buffer or pass a plain byte slice.
- **Metadata blocks** accompany every operation with timing, interface, protocol, and user-extensible fields (e.g., TLS session IDs). Kraken uses these fields to enrich its telemetry without touching module code.

Usage Patterns

- **Kraken transports** — Campaign files specify `tcp`, `tls`, `udp`, or `dtls`. Trident instantiates the right conduit and exposes it through the module ABI.
- **Logging & diagnostics** — Wrap any conduit with the logging decorator to get structured traces (dial duration, payload sizes, remote endpoints) without instrumenting application code.

Testing & Support

- `go test ./trident/...` runs the unit suite covering stream/datagram conduits plus integration tests for TLS/DTLS handshakes.

Roadmap

- ☐ IP and Ethernet protocol support and testing
- ☐ QUIC and SCTP conduits.

- ☐ HTTP(S) helper conduits for quick control-channel testing.
- ☐ WebSocket and HTTP/3 stacks.
- ☐ Connection pooling with built-in metrics hooks.
- ☐ io_uring backend for Linux hosts.