□ Kraken System Requisites

Architecture Overview

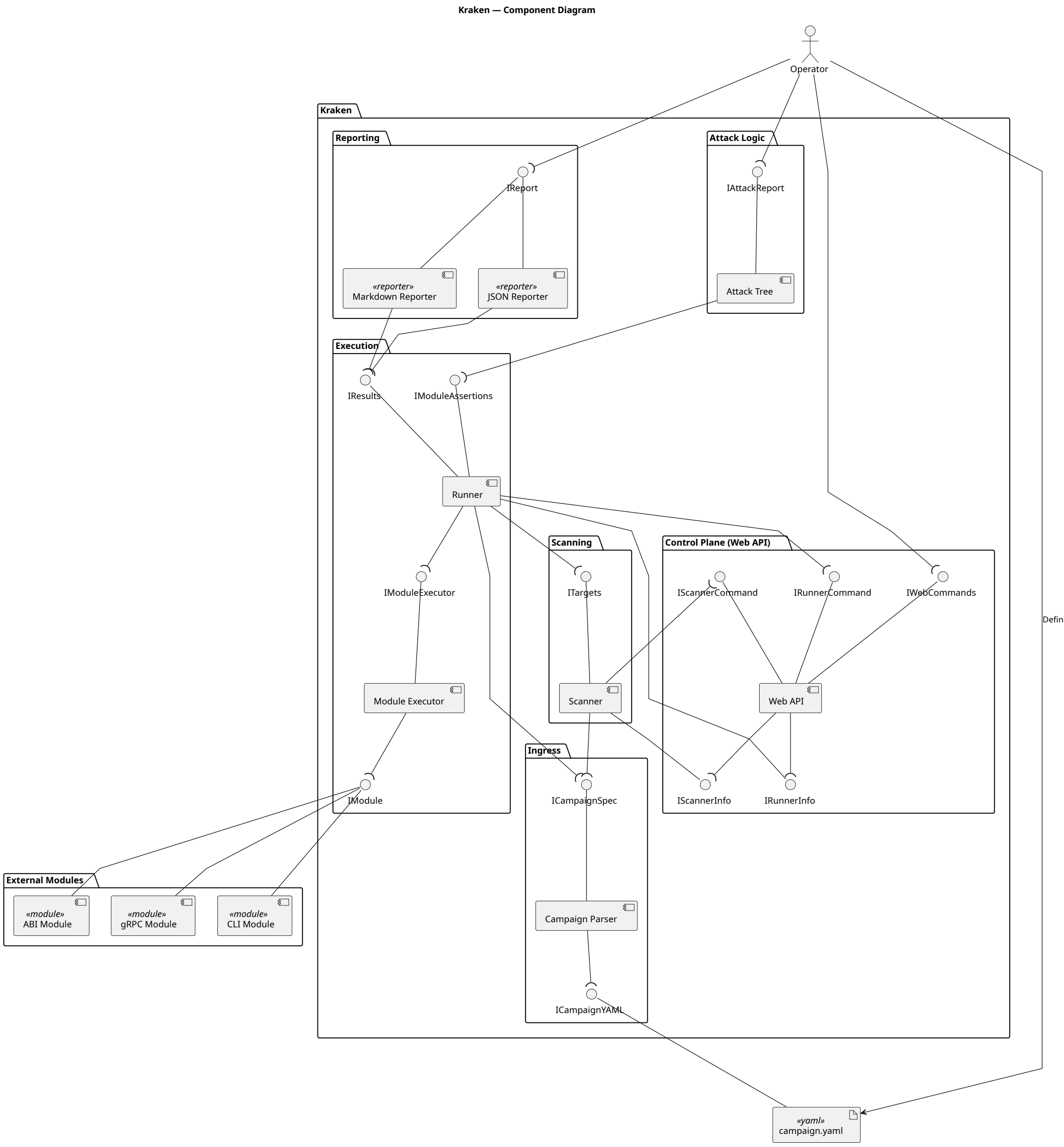

Figure 1: architecture

1. High-Level Requirements

1.1 Core Functionality

- **HL-F1** — Kraken shall execute security assessment campaigns against specified network targets.
- **HL-F2** — Kraken shall be extensible through modules to support various transport and network protocols.
- **HL-F3** — Kraken shall discover and scan network targets to identify services and potential vulnerabilities.
- **HL-F4** — Kraken shall aggregate results from its components and generate reports in human-readable and machine-readable formats.

1.2 User Interaction

- **HL-U1** — Kraken shall provide a command-line interface (CLI) for campaign execution and configuration.
- **HL-U2** — Kraken shall provide a web-based graphical interface for real-time monitoring and control of campaigns.
- **HL-U3** — Campaigns shall be defined in a structured format (e.g., YAML) that specifies modules and targets.

1.3 System Qualities

- **HL-Q1** — Kraken shall be delivered as a single, fast, and lightweight binary executable (e.g., final binary size < 50MB).
- **HL-Q2** — The system shall be modular, with clear separation of concerns between its subsystems.
- **HL-Q3** — The system shall be resilient, with a recovery mechanism to handle module crashes without halting the entire campaign.
- **HL-Q4** — The system shall be performant, executing modules concurrently to minimize campaign duration.

---

2. Low-Level Requirements

These requirements provide detailed specifications for the implementation of each component.

2.1 General

- **LL-G1** — The `cli` interface shall be composed of 4 commands:
  1. `--campaign`: specify campaign path to load the campaign from
  2. `--cidrs` (optional): specify cidr/host to target in the campaign, can be specified inside the campaign.
  3. `--out`(default: `./results`): specify where to output campaign's output
  4. `--help`: print the help to use the cli interface of the program.

2.2 Runner Component (KRK-RUN)

2.2.1 Architectural Requirements

- **KRK-RUN-A1** — The Runner shall be a core subsystem of Kraken responsible for executing modules defined in campaigns.
- **KRK-RUN-A2** — The Runner shall communicate via the Trident conduits (transport abstraction).
- **KRK-RUN-A3** — The Runner shall provide an internal interface for module scheduling, execution, and lifecycle management.
- **KRK-RUN-A4** — The Runner shall isolate each module execution to prevent cross-failure.

2.2.2 Functional Requirements

- **KRK-RUN-F1** — The Runner shall not crash if a module crashes.
- **KRK-RUN-F2** — The Runner shall execute modules in parallel.
- **KRK-RUN-F3** — The Runner shall support a configurable maximum number of parallel module executions.
- **KRK-RUN-F4** — The Runner shall execute modules targeting different targets in parallel.
- **KRK-RUN-F5** — The Runner shall be configurable from the campaign YAML file.
- **KRK-RUN-F6** — The Runner shall collect module outputs for the Reporter.
- **KRK-RUN-F7** — The Runner shall handle configurable retry and timeout logic during module execution.

2.2.3 Non-Functional Requirements

- **KRK-RUN-N1** — The Runner shall efficiently use system resources during concurrent execution (e.g., CPU usage should not exceed 80% for a sustained period on benchmark hardware).
- **KRK-RUN-N2** — The Runner shall ensure stability under high concurrency (e.g., complete a benchmark campaign with 1000 parallel modules without crashing).
- **KRK-RUN-N3** — The Runner shall produce structured logs for every module execution.
- **KRK-RUN-N4** — The Runner shall recover gracefully from transient system or network errors.

2.3 Scanner Component (KRK-SCN)

2.3.1 Architectural Requirements

- **KRK-SCN-A1** — The Scanner shall serve as Kraken's discovery subsystem.
- **KRK-SCN-A2** — The Scanner shall integrate with Nmap for IP-layer discovery and scanning.
- **KRK-SCN-A3** — The Scanner shall expose its results to the Runner and Reporter through a standardized data schema.
- **KRK-SCN-A4** — The Scanner shall support modular extensions for future discovery engines.

2.3.2 Functional Requirements

- **KRK-SCN-F1** — The Scanner shall perform network reconnaissance as the first step of a campaign.
- **KRK-SCN-F2** — The Scanner shall use Nmap for IP-layer scanning (TCP/UDP ports).
- **KRK-SCN-F3** — The Scanner shall identify targets at the L2 layer.

- **KRK-SCN-F4** — The Scanner shall assign classification tags to identified targets.
- **KRK-SCN-F5** — The Scanner shall be configurable via YAML parameters.
- **KRK-SCN-F6** — The Scanner shall use non-invasive and non-aggressive options by default (e.g., avoiding scans or scripts considered intrusive).
- **KRK-SCN-F7** — The Scanner shall support both IPv4 and IPv6 scanning.

2.3.3 Non-Functional Requirements

- **KRK-SCN-N1** — The Scanner shall minimize network footprint during reconnaissance (e.g., default scan for a /24 subnet should generate < 1GB of traffic).
- **KRK-SCN-N2** — The Scanner shall ensure accuracy of service and protocol identification.
- **KRK-SCN-N3** — The Scanner shall be extensible for integration with other discovery tools.
- **KRK-SCN-N4** — The Scanner shall complete discovery within a configurable timeout per target (e.g., default timeout of 5 minutes per host).

2.4 Reporter Component (KRK-REP)

2.4.1 Architectural Requirements

- **KRK-REP-A1** — The Reporter shall serve as the result aggregation and reporting subsystem.
- **KRK-REP-A2** — The Reporter shall consume outputs from the Runner and Scanner through defined interfaces.
- **KRK-REP-A3** — The Reporter shall support multiple pluggable output formats.
- **KRK-REP-A4** — The Reporter shall write all reports under the campaign's `result` directory.

2.4.2 Functional Requirements

- **KRK-REP-F1** — The Reporter shall aggregate findings from all modules and scanners.
- **KRK-REP-F2** — The Reporter shall generate campaign-level summaries (metadata, duration, target counts).
- **KRK-REP-F3** — The Reporter shall produce reports in three formats:
  1. **Markdown** — for human readability
  2. **JSON** — for machine readability.
  3. **Interactive HTML** — for viewing in a browser.
- **KRK-REP-F4** — The Reporter shall include severity levels and result statuses (success/failure).

2.4.3 Non-Functional Requirements

- **KRK-REP-N1** — The Reporter shall generate reports with minimal delay after findings are available (e.g., a 1000-finding JSON report should be generated in < 10 seconds).
- **KRK-REP-N2** — The Reporter shall ensure report files are deterministic and reproducible.
- **KRK-REP-N3** — The Reporter shall maintain consistent schema across versions for backward compatibility.