Siren

Siren is a transparent man-in-the-middle agent that lives almost entirely inside the Linux kernel. It pairs eBPF programs (XDP, TC and socket hooks) with optional TLS function probes to watch network conversations, apply lightweight edits and feed structured evidence back to analysts.

Architecture

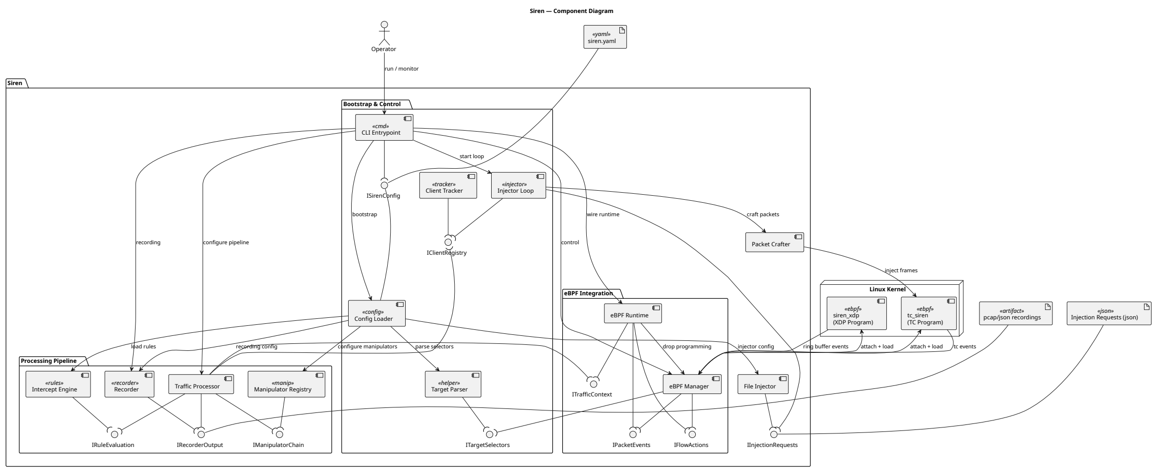

Figure 1: architecture

Traffic first touches the XDP program to decide whether the packet should be kept, mirrored or instantly dropped. Flows that need move to a TC (traffic Control) program where headers or payloads can be modified. Socket-level hooks observe connection metadata (latency, retransmits) while optional TLS uprobes copy plaintext buffers before they are encrypted. Everything is streamed to the userspace runtime which evaluates the YAML rules and feeds back decisions through BPF maps.

Playbook

Siren is configured with a single YAML file called a *playbook*. The playbook describes what to watch, how to record it and which actions should be taken.

- **General options**: describe the scenario and keep track of the playbook version.

  ```yaml
  name: "factory-floor-mitm"
  version: "1.2.0"
  description: |
      Inline inspection for the assembly VLAN, drop dangerous telnet
      commands and mask MQTT credentials.
  ```

- **Network attachment**: decide where the eBPF programs run and which helpers are enabled. `xdp` provides line-rate filtering, `tc` handles edits, and `socket` emits per-flow events. TLS hooks attach to userland libraries so you can see decrypted buffers when needed.

```yaml
ebpf:
    interface: eth0
    programs:
        xdp: ingress
        tc: true
        socket: true
    drop_action_duration: 10s
    targets:
        - "ip:192.0.2.10"
        - "ip_port:192.0.2.30:502"
        - "mac:aa:bb:cc:dd:ee:ff"
tls:
    enabled: true
    providers:
        - library: /usr/lib/libssl.so.3
          read: SSL_read
          write: SSL_write
```

- **Recording & telemetry**: control how evidence is stored. Siren can generate only pcap for now.

```yaml
recording:
    enabled: true
    format: pcap
    output: /var/tmp/siren-floor-a.pcap
```

- **Rulebook**: list match conditions and the action to run when a packet or flow matches. Matchers include payload regex, direction, and interface labels. Actions map to the pipeline: `drop` happens in XDP, `rewrite` in TC, `throttle`/`mirror` through socket helpers, and `inject` collaborates with userspace manipulators.

```yaml
rules:
    - name: block-dangerous-telnet
      match:
          payload_regex: "USER root"
          l4_protocol: tcp
          port: 23
      action:
          type: drop

    - name: soften-modbus
      match:
          target_selector: "ip_port:192.0.2.30:502"
      action:
          type: throttle
          rate: 40pps
          duration: 20s

    - name: hide-mqtt-credentials
      match:
          payload_regex: "username:"
```

```yaml
        action:
            type: rewrite
            payload_patch: "username: <redacted>"

      - name: inject-smtp-banner
        match:
            direction: server
            payload_regex: "^EHLO"
        action:
            type: inject
            template: "smtp_banner.txt"
```

If the `targets` list is empty the playbook applies to every frame on the interface. TLS providers are optional; when omitted Siren still operates, but payload rules will only see ciphertext.

Pipelines

XDP pipeline

Runs at the earliest point in the NIC driver. It samples packets into the ring buffer, enforces drop decisions, and tags flows that need extra processing.

TC pipeline

TC is where Siren rewrites headers, pads payloads or injects canned responses. Because the packets never leave the kernel, edits are fast and opaque to endpoints.

TLS hooks

Optional probes on userland TLS libraries expose the plaintext seen by `SSL_read`/`SSL_write`. Siren only touches the libraries listed in the playbook, if a server is not using OpenSSl but BoringSSL then it would not be able to inspect the encrypted traffic.

Running Siren

```
sudo ./siren -config siren/config/playbook.yaml
```

The CLI attaches the selected eBPF programs, loads TLS hooks when requested and starts streaming events. If the NIC refuses XDP, run `ethtool -K <iface> rxvlan off gro off` and try again. Outputs land in the files you specified in the playbook and can be opened directly in Wireshark.

Building the eBPF objects

First the kernel eBPF programs needs to be compiled using the following command.

```
go generate ./siren/ebpf
```

> [!NOTE] Be sure to have LLVM, linux headers, ebpf headers and clang installed.

Then simply using `go build ./siren` should build the program.

Finally to attach XDP/TC programs there is the need of network capabilities or root privileges.

```
sudo ./siren -config siren/config/config.yaml
```