Stream Expectation & Scheduler Routing Specification

1. Purpose

This document defines how **White Venom** classifies, profiles, and routes incoming event streams in a **reactive, security-aware scheduler architecture**.

The goal is **not filtering**, **not blocking**, and **not signature-based detection**, but **expectation-based stream handling** with deterministic fallback to a NULL scheduler.

> Analogy: *"If bad air comes in, we open the window and let it out."*

---

2. Core Principles

1. **Security is not a state** – it is continuous regulation.
2. **Events are not equal** – streams have intent, shape, and behavior.
3. **No hard denial** – only routing and attenuation.
4. **No feedback to the source** – avoid attacker learning loops.
5. **Schedulers are isolated domains**, not shared queues.

---

3. Stream Taxonomy

3.1 DataType

Defines what kind of data a stream is expected to carry.

```
TEXT        – human-readable configuration, commands
JSON        – structured configuration or metadata
METRIC      – numeric telemetry, audit counters
BINARY      – opaque data blobs
UNKNOWN     – undecidable or malformed
```

---

4. Stream Expectation Model

Each stream source is bound to a **StreamExpectation** contract.

```
struct StreamExpectation {
    DataType expectedType;
    size_t   maxPayloadSize;
    uint32_t maxRatePerSec;
    bool     allowBurst;
};
```

This is **not validation**. It is a *behavioral envelope*.

---

5. Stream Profiles

Each stream has **two operational profiles**:

5.1 NORMAL Profile

- Human-scale interaction
- Predictable event rate
- Minimal burst tolerance

5.2 HIGH Profile

Activated during: - system boot - recovery - elevated threat posture - internal reconfiguration

```
struct StreamProfileConfig {
    StreamExpectation normal;
    StreamExpectation high;
};
```

---

6. Automatic Profile Escalation

The system dynamically escalates profiles based on observed metrics.

```
NORMAL → HIGH → NULL
```

Escalation criteria include: - sustained rate exceedance - payload size inflation - malformed content ratio - scheduler pressure indicators

There is **no de-escalation feedback** to the event source.

---

7. Stream Probe (Behavioral Analysis)

Before routing, each stream is sampled by a lightweight probe.

```
struct StreamProbeResult {
    DataType detectedType;
    size_t   avgPayloadSize;
    uint32_t rate;
    bool     malformed;
};
```

The probe: - does **not parse deeply** - does **not allocate heavily** - does **not block schedulers**

---

8. Scheduler Routing Logic

Routing is deterministic and auditable.

```
MATCHES NORMAL   → HighPriorityScheduler
MATCHES HIGH     → LowPriorityScheduler
MATCHES NONE     → NullScheduler
```

Example:

- TEXT expected
- BINARY observed
- HIGH exceeded

→ Routed to **NULL Scheduler**

---

9. NULL Scheduler Semantics

The NULL Scheduler:

- accepts events
- drops them immediately
- does not log per-event
- does not respond
- optionally emits **aggregated metrics only**

This prevents: - DoS amplification - timing inference - feedback-based probing

---

10. Security Posture Outcome

This model ensures:

- attacker ignorance of internal state
- bounded resource consumption
- clean separation of concerns
- predictable system behavior under stress

"Bad input does not break the system — it is simply ventilated out."

---

11. Design Philosophy

This architecture is inspired by:

- Reactive Streams (RX)
- Operating system schedulers
- Biological nervous systems

It favors **calm systems over loud defenses**.

---

12. Closing Note

This specification is intended to be: - readable by engineers - enforceable in code - defensible in audits

Security here is **quiet, boring, and effective**.