SPEC-03: Sentry Module

1. Background

The **Sentry** module provides proactive monitoring, alerting, and incident management for the Hearthlink ecosystem. It ingests metrics, logs, and security events from Vault, Core Services, Synapse, and infrastructure components; detects anomalies via rules and ML detectors; and surfaces alerts through a rich "Batcave"-style dashboard with multiple live system feeds and remediation hooks.

2. Requirements (MoSCoW)

Must have

- Real-time ingestion (<2s latency) of metrics/logs via Kafka or equivalent
- Pluggable anomaly-detection engine (threshold, statistical, ML)
- RBAC-aware alert routing with email, Slack, PagerDuty integrations
- Configurable escalation policies per service and severity
- Durable event store with 90-day retention and automated cycling to avoid bloat

Should have

- Automated remediation hooks (e.g., scale pods, restart services) via webhooks
- Correlated incident timelines with links to message IDs
- Dashboard drill-down from overview to event details

Could have

- Multi-tenant dashboard segmentation
- Predictive fatigue prevention (alert suppression logic)

Won't have (this increment)

- Mobile push notifications
- ChatOps/bot integrations

3. Method

3.1 Architecture Diagram

```
@startuml
package "Sentry Cluster" {
  [Ingestion API] --> [Event Queue]
  [Rule Engine] --> [Event Queue]
  [Rule Engine] --> [Alert Dispatcher]
  [Event Store] <-- [Ingestion API]
}
package "Consumers" {
  [Dashboard UI]
  [Automation Hooks]
  [Notification Channels]
}
[Consumers] --> [Alert Dispatcher]
@enduml
```

3.2 Data Schema

```
@startuml
table EventRecord {
  + event_id    : UUID [PK]
  + source      : VARCHAR
  + type        : VARCHAR
  + payload     : JSON
  + severity    : ENUM('INFO','WARN','ERROR','CRITICAL')
  + timestamp   : TIMESTAMP
}

table Alert {
  + alert_id    : UUID [PK]
  + event_id    : UUID [FK]
  + rule_id     : UUID
  + state       : ENUM('OPEN','ACK','RESOLVED')
  + assigned_to : VARCHAR
  + created_at  : TIMESTAMP
  + updated_at  : TIMESTAMP
}

table IncidentTimeline {
  + timeline_id : UUID [PK]
  + alert_id    : UUID [FK]
  + entry       : TEXT
  + timestamp   : TIMESTAMP
}
@enduml
```

4. UI Components & Wireframes

4.1 Batcave Dashboard Overview (Full-Screen)

```
+==================================================================================+
| Sentry Batcave
```

```
Dashboard                                                             |
| [Refresh] [Settings] [Help]
[User]                                                   |
|-------------------------------------------------------------------------------------|
| | Live Metrics Feed      | Service Health Map     | Alert Summary
Panel           |
| | (Scrolling logs)       | (Heatmap over topology)| (Counts by
severity)         |
|-------------------------------------------------------------------------------------|
| | Incident Timeline      | Rules & Policies Panel | Automation Hooks
Panel        |
| | (Chronological view)   | (List + create/edit)   | (Retry, scale,
notify)       |
|-------------------------------------------------------------------------------------|
+=====================================================================================+
```

Component Mapping

Component	Function	Data/API Call
LiveMetricsFeed	Real-time log/metric stream	`GET /v1/events/stream` (WebSocket)
ServiceHealthMap	Visual heatmap of service statuses	`GET /v1/health/map`
AlertSummaryPanel	Aggregated alert counts by severity and service	`GET /v1/alerts/summary`
IncidentTimelineView	Chronological incident entries	`GET /v1/incidents/timeline?alert={id}`
RulesPoliciesPanel	List, create, update anomaly-detection rules	`GET/POST/PUT/DELETE /v1/alerts/policies`
AutomationHooksPanel	Buttons to trigger remediation actions	`POST /v1/alerts/{id}/remediate`
RefreshButton	Reload all panels	Triggers all GET endpoints
SettingsButton	Open Sentry module settings modal	N/A
HelpButton	Opens documentation	External link
UserMenu	Profile/logout	N/A

4.2 Alert List Panel

```
+--------------------------------------------------------------+
| Current Alerts                                               |
| [Filter: Service ▼] [Severity ▼] [Search] [Acknowledge All] |
|-------------------------------------------------------------|
| | AlertID | Service | Severity | Status | Assigned | Age |  |
|-------------------------------------------------------------|
+-------------------------------------------------------------+
```

Component	Function	Data/API Call
AlertsListTable	Display open alerts with filters	`GET /v1/alerts?filter...`
FilterDropdowns	Filter by service/severity	Client-side filtering + API query parameters
AcknowledgeButton	Acknowledge selected alert	`POST /v1/alerts/{id}/ack`
ResolveButton	Resolve selected alert	`POST /v1/alerts/{id}/resolve`
AcknowledgeAllButton	Bulk ack all alerts	`POST /v1/alerts/ackAll`

4.3 Rules & Policies Editor

```
+-------------------------------------------------------------+
| Rules & Policies                                          X |
| [New Rule] [Import] [Export]                               |
|-------------------------------------------------------------|
| | RuleID | Name | Condition | Action | Enabled | Created   |
|-------------------------------------------------------------|
| Rule Detail Pane (below): JSON editor with live validation |
+-------------------------------------------------------------+
```

Component	Function	Data/API Call
RulesTable	List existing rules	`GET /v1/alerts/policies`
NewRuleButton	Open modal to define a new rule	N/A
ImportRulesButton	Bulk import rules	`POST /v1/alerts/policies/import`
ExportRulesButton	Export rules to JSON	`GET /v1/alerts/policies/export`
RuleDetailEditor	JSON editor for rule definition	`GET/PUT /v1/alerts/policies/{id}`
EnableToggle	Enable/disable a rule	`PUT /v1/alerts/policies/{id}`

4.4 Incident Timeline Full View

```
+-------------------------------------------------------------+
| Incident Timeline                                          |
| [Back to Dashboard] [Filter: AlertID ▼]                    |
|-------------------------------------------------------------|
| | Timestamp | Actor | Entry Description                        |
|-------------------------------------------------------------|
+-------------------------------------------------------------+
```

Component	Function	Data/API Call
TimelineTable	List timeline entries	`GET /v1/incidents/timeline?alert={id}`
BackButton	Navigate back to dashboard	N/A
FilterDropdown	Filter by alert ID	Client-side or `?alert=` query

5. API Endpoints

Method	Path	Description	Auth Scope
POST	/v1/events	Ingest new event	`sentry.ingest`
GET	/v1/events/stream	Subscribe to live event stream	`sentry.read`
GET	/v1/health/map	Fetch service health map data	`sentry.read`
GET	/v1/alerts	List current alerts	`sentry.alert.read`
POST	/v1/alerts/{id}/ack	Acknowledge alert	`sentry.alert.write`
POST	/v1/alerts/{id}/resolve	Resolve alert	`sentry.alert.write`
GET	/v1/alerts/policies	List anomaly rules	`sentry.policy.read`
POST	/v1/alerts/policies	Create new rule	`sentry.policy.write`
GET	/v1/incidents/timeline	Fetch incident timeline entries	`sentry.alert.read`
POST	/v1/alerts/policies/import	Import rules from JSON	`sentry.policy.write`
GET	/v1/alerts/policies/export	Export rules as JSON	`sentry.policy.read`

6. Implementation

1. Deploy Kafka and schema registry; setup Ingestion API
2. Build Rule Engine microservice (Flink/KStreams)
3. Implement Alert Dispatcher and Notification adapters
4. Develop Batcave Dashboard in React + Recharts
5. Integrate WebSocket for live streams, REST for controls
6. Enforce RBAC on all endpoints; secure mTLS
7. Add audit logging for all rule changes and alert actions
8. Load-test >10k events/sec; fault-injection scenarios

7. Milestones

Milestone	Timeline	Owner
Ingestion & Event Store Setup	Week 1	DevOps Lead
Rule Engine & Policies MVP	Week 2–3	Data Eng Team
Alert Dispatcher & Notifications	Week 4	Backend Team
Dashboard & Live Feed Integration	Week 5–6	Frontend Team
RBAC & Security Hardening	Week 7	Security Team
Performance & Reliability Testing	Week 8–9	QA Team

8. Gathering Results

- Ingestion latency <2s under load
- Dashboard live stream refresh <1s
- Rule accuracy >95% with <5% false triggers
- Alert ack/resolve workflows <1s median
- RBAC misuse incidents = 0 in penetration tests

9. References & Dependencies

- **Integration Blueprints**: appendix_b_integration_blueprints.md
- **DevOps Guide**: _DEVELOPMENT_OPERATIONS_GUIDE.md
- **Security Policies**: VOICE_ACCESS_POLICY.md
- **SOPs**: SOP_Retrospective_Cycle.md for incident workflows

Need Professional Help in Developing Your Architecture?

Please contact me at sammuti.com :)