Project: Event Processing System – Original Code Review
Issue Analysis Report

Hamza Zarour

1) All issues you identified in the original code

EventProcessor (original code)

- **Contains one large prosess() method that handles all responsibilities.**
- **Performs validation, notification, transformation, persistence, and type-specific logic in one place.**
- **Uses multiple if statements for encrypt, compress, and addMetadata.**
- **Uses a large if/else block based on event type strings "USER", "SYSTEM", "SECURITY"**
- **Directly depends on Databese, Dashboard, and Logger.**
- **Contains hard-coded security behavior inside the processor.**
- **Notifies dashboard and logger before generating the event id.**

Dashboard / Logger

- **Called directly from inside EventProcessor.**
- **Cannot be added or removed without modifying the processor**

Event (data model)

- **Uses multiple boolean flags that increase logic complexity**
- **Allows metadata to be missing while addMetadata is enabled.**

2) Why each issue is problematic

EventProcessor

- **A large method is hard to read, test, and maintain.**
- **Mixing many responsibilities increases the risk of bugs**
- **Multiple if statements reduce scalability and readability.**
- **Type-based if/else violates the Open/Closed Principle**
- **Tight coupling prevents easy reuse and extension.**
- **Hard-coded logic makes changes risky and error-prone.**
- **Incorrect ordering may cause logs and metrics to miss the event id.**

Dashboard / Logger

- **Adding new modules requires changing core processing code.**
- **This increases maintenance efforts and reduces flexibility.**

Event

- **Boolean flags push complexity into the processor.**
- **Missing metadata may produce incorrect output formatting.**

3. How the Refactoring Solves the Problems

Observer Pattern

- **Separates notification logic from EventProcessor.**
- **Dashboard and Logger are implemented as listeners.**
- **New modules can be added without modifying the processor.**

Decorator Pattern

- **Removes transformation if statements from the processor.**
- **Each transformation is implemented as a separate decorator.**
- **Allows flexible combination of transformations.**

Strategy Pattern

- **Removes type-based if-else logic from EventProcessor.**
- **Each event type has its own strategy class.**
- **Adding new event types does not require modifying the processor.**

Processing order fix

- **The new flow ensures correct execution order:**
  validate , transform , generate id , save , notify , execute strategy
- **Ensures observers always receive correct and complete event data.**