22AIE111-OBJECT ORIENTED PROGRAMMING IN JAVA

&

22AIE112-DATA STRUCTURES & ALGORITHMS-1

Cybersecurity Incident Response System

Nimmagadda Kesav Satya Sai	CB.SC.U4AIE24236
Rupanshi Sangwan	CB.SC.U4AIE24262
Vemula Poornachandra	CB.SC.U4AIE24258
Nakka Saampotth Maddileti	CB.SC.U4AIE24233

Introduction

- With the increasing number of cyber threats, organizations require an efficient incident response system.
- The Cybersecurity Incident Response System (CIRS) is designed to enhance an organization's ability to effectively detect, respond to, and mitigate cyber threats.
- The system leverages a combination of data structures to prioritize, track, and analyze security incidents.
- It streamlines the incident resolution process, ensuring faster and more effective responses.

Importance of Incident Response Planning

Timely and Organized Response

Having a well-defined incident response plan ensures that your team can act quickly and effectively when a security breach or incident occurs.

Clear Communication

Incident response planning sets out clear communication strategies, within the response team. This ensures that important information is shared accurately, reducing miscommunication during high-pressure situations.

Identification of Weaknesses

The process of preparing and testing an incident response plan helps identify vulnerabilities in your systems that could potentially be exploited in an attack.

Continuous Improvement

The incident response plan allows for post-incident analysis to understand what worked well and what could be improved.

Problem Statement

Challenges in Cyber Incident Response:

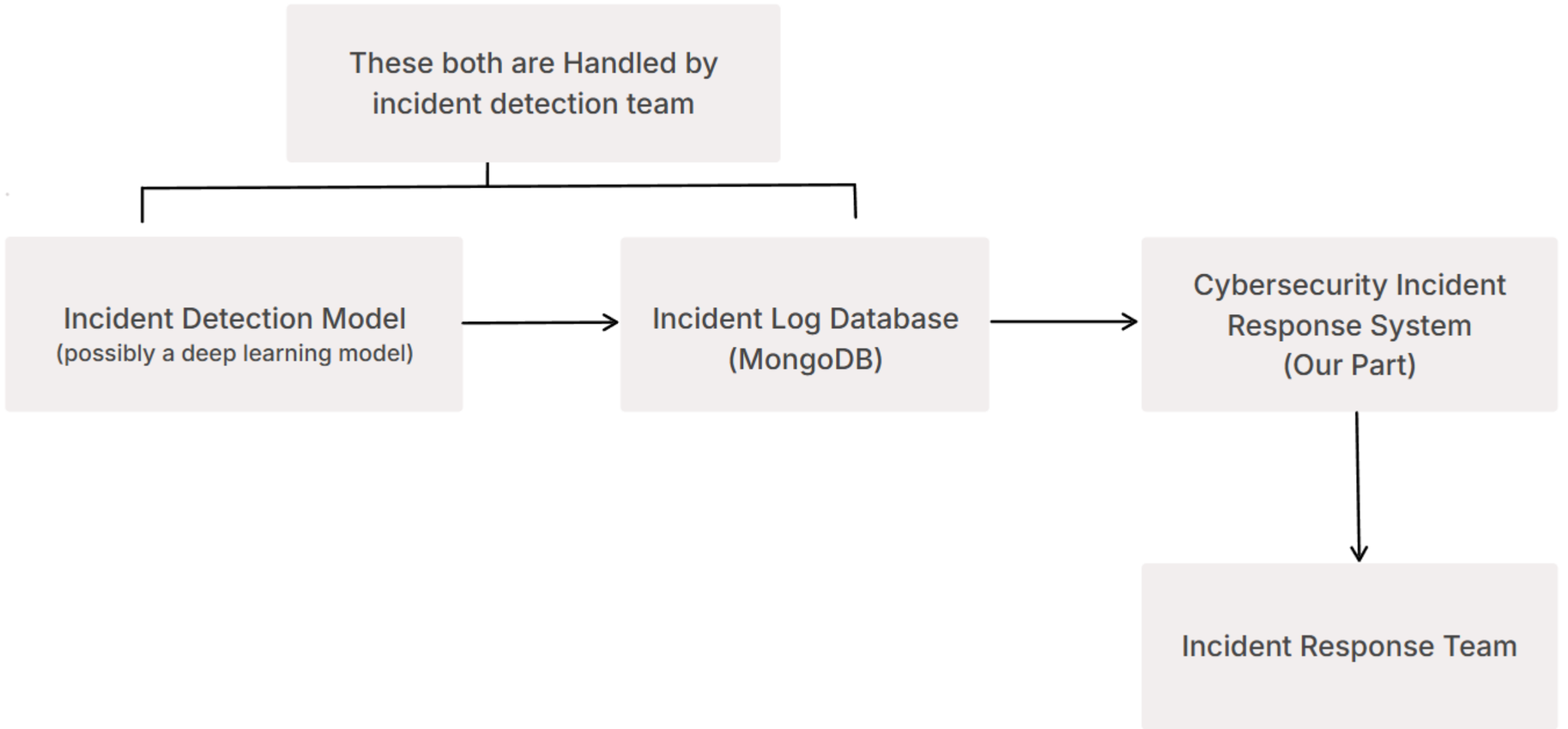
- High volume of incidents makes manual handling inefficient.
- Lack of structured data leads to delays in incident resolution.
- No standardized response procedures, making decision-making inconsistent.
- Prioritization issues in handling critical threats.

These points highlight the need for a structured and automated system to improve incident handling and response efficiency.

Objectives

- Store, classify, and organize cyber incidents in a structured format.
- Read data from a MongoDB incident log database and present it in a user-friendly interface.
- Maintain a priority queue based on severity to resolve critical incidents first.
- Provide response teams with pre-defined customized checklists based on incident type & subtype.

Interpretation



Involvement of Data Structures

Linked List:
Used to store all incidents in the order they were retrieved from database.
We used Linked List because of the Fast insertions possible in Linked lists and the way it preserves chronological order.

Priority Queue:
Stores incidents in a sorted manner in a separate log based on severity.
This data structure helps in efficient retrieval of highest-severity incidents.

Hash Map:
Stores predefined checklists for each type of incident and its sub-type.
This data structure helps in looking up for response checklists.

Why is Linked List preferred over an Array?

- **Easy to Grow or Shrink**: Linked lists can change size without moving everything around. Adding at the beginning or end takes O(1) time, while arrays might need O(n) time to resize.
- **Faster Insertions in the Middle**: Adding an element in the middle of a linked list takes O(1) time once you find the spot, but arrays need O(n) time because you must shift elements.
- **Faster Deletions in the Middle**: Removing from the middle of a linked list is O(1) after finding the location, while arrays need O(n) time to close the gap.
- **No Wasted Space**: Linked lists use exactly the memory they need, while arrays might reserve extra unused space.
- **No need for initial size declaration**: Unlike arrays, linked lists don't require specifying a size at creation, making them ideal when the data size is unknown.

Time Complexity of Priority Queue Operations

- **Insertion in Priority Queue**: O(log n) - New elements are added and positioned correctly through heap adjustment
- **Deletion in Priority Queue**: O(log n) - Highest priority element is removed from the root, followed by heap restructuring
- **Extract Min/Max**: O(log n) - Retrieving and removing the highest priority element requires rebalancing the heap
- **Get Min/Max (Peek)**: O(1) - Accessing the highest priority element without removal
- **Decrease/Increase Key**: O(log n) - Changing an element's priority requires repositioning in the heap

Implementation of OOP Concepts

"Incident"(Entity Class):

- Attributes: id, description, type, subtype, severity
- Implements Comparable<incident> for prioritizing incidents based on severity.
- **Responsibility**: Represents the structure of an incident.

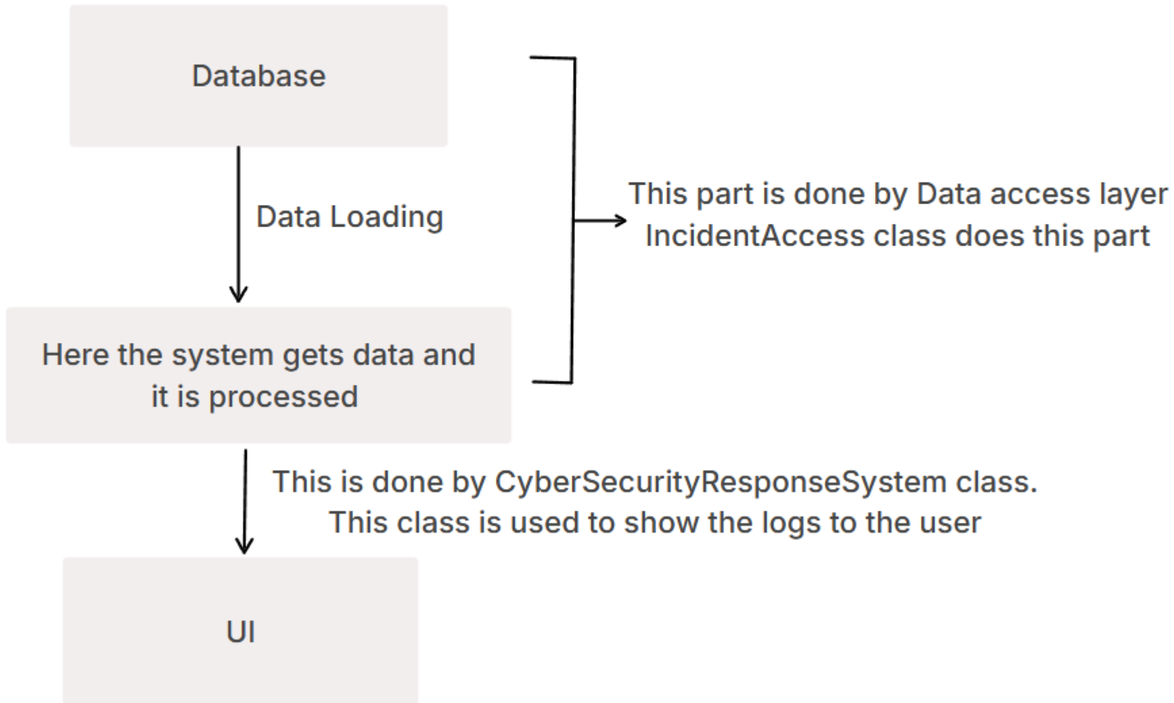
"IncidentManager" (Data Access Layer):

- Connects to MongoDB.
- Stores incidents chronologically in a Linked List and sorted by severity in a Prirority Queue.
- **Responsibility**: Handles fetching and storing of incident data.

"CyberSecurityResponseSystem" (UI & Business Logic):

- Implements Java Swing for GUI.
- Displays incident logs and their priorities.
- Provides checklists based on incident type for the response team.
- **Responsibility**: Manages user interaction and system operations.

Interpretation of OOPs usage



Data

- Incidents are loaded from MongoDB
- The database simulates the incidents detected by the model
- For the project instance we are preparing a simulated database having incidents in a pre-defined format
- Each incident follows a predefined format (ID, Type, Subtype, Description, Severity).

Framework

- As this is a real-world application, we choose object-oriented procedure for re-usability. So we will be using java for this project
- MongoDB is used to store the incident logs generated by the AI model (fake data for the project instance). Java interacts with MongoDB to read and structure data.
- Java Swing is used for building the Graphical user interface

Thank You!