**RHYM – Mini Project**

### **Use Cases**

#### Use Case 1: View Cybersecurity Information

- \*\*Actor\*\*: User

- \*\*Description\*\*: A user views the latest cybersecurity information, including threat actors, techniques, on the web portal.

- \*\*Preconditions\*\*: The user has access to a web browser.

- \*\*Postconditions\*\*: The user can see the latest data on threat actors, techniques.

- \*\*Main Flow\*\*:

1. The user opens the web portal.

2. The system displays the latest threat actors, techniques.

3. The user browses through the information.

#### Use Case 2: Scrape Cybersecurity Data

- \*\*Actor\*\*: System

- \*\*Description\*\*: The system scrapes cybersecurity data from predefined sources.

- \*\*Preconditions\*\*: The sources are accessible and the scraping script is correctly configured.

- \*\*Postconditions\*\*: The latest data is saved in CSV files.

- \*\*Main Flow\*\*:

1. The system initiates the scraping process.

2. The system retrieves data from the sources.

3. The system parses and processes the data.

4. The system saves the data into CSV files.

#### Use Case 3: Automate Data Scraping

- \*\*Actor\*\*: System Administrator

- \*\*Description\*\*: The system administrator sets up an automated task to scrape data at regular intervals.

- \*\*Preconditions\*\*: The system administrator has access to the server.

- \*\*Postconditions\*\*: The data is automatically scraped and updated.

- \*\*Main Flow\*\*:

1. The system administrator configures a task scheduler.

2. The task scheduler runs the scraping script at predefined intervals.

3. The system performs the scraping process as described in Use Case 2.

**### Product Backlog**

#### Epic 1: Web Scraping and Data Storage

- \*\*User Story 1.1\*\*: As a developer, I want to scrape data from cybersecurity websites so that I can gather information on threat actors, techniques.

- \*\*Acceptance Criteria\*\*:

- Data is successfully scraped from multiple sources.

- Data is parsed and saved into structured formats (CSV files).

- \*\*User Story 1.2\*\*: As a developer, I want to handle errors and exceptions during scraping so that the system is robust and reliable.

- \*\*Acceptance Criteria\*\*:

- Errors are logged.

- Scraping continues or retries in case of transient errors.

#### Epic 2: Web Portal Development

- \*\*User Story 2.1\*\*: As a user, I want to view the latest threat actors on the web portal so that I can stay informed about potential threats.

- \*\*Acceptance Criteria\*\*:

- Threat actors data is displayed in a tabular format.

- \*\*User Story 2.2\*\*: As a user, I want to view the latest techniques on the web portal so that I can understand the methods used by threat actors.

- \*\*Acceptance Criteria\*\*:

- Techniques data is displayed in a tabular format.

- \*\*User Story 2.3\*\*: As a developer, I want to style the web portal using CSS frameworks so that it looks professional and is easy to navigate.

- \*\*Acceptance Criteria\*\*:

- The web portal is styled with a responsive design using a CSS framework (e.g., Bootstrap).

#### Epic 3: Automation and Scheduling

- \*\*User Story 3.1\*\*: As a system administrator, I want to automate the data scraping process so that the information is kept up to date without manual intervention.

- \*\*Acceptance Criteria\*\*:

- A task scheduler is set up to run the scraping script at regular intervals (e.g., daily, weekly).

- \*\*User Story 3.2\*\*: As a system administrator, I want to receive notifications in case of scraping failures so that I can take corrective action.

- \*\*Acceptance Criteria\*\*:

- Notifications are sent via email or a messaging service when scraping fails.

#### Epic 4: Database Integration (Optional Enhancement)

- \*\*User Story 4.1\*\*: As a developer, I want to store the scraped data in a database so that it can be managed and queried efficiently.

- \*\*Acceptance Criteria\*\*:

- Scraped data is saved in a database (e.g., SQLite, PostgreSQL).

- \*\*User Story 4.2\*\*: As a developer, I want to retrieve data from the database to display on the web portal so that the portal shows the latest information.

- \*\*Acceptance Criteria\*\*:

- The web portal retrieves data from the database.

**### Task Breakdown**

#### Sprint 1: Basic Scraping and Data Storage

1. Set up the project environment.

2. Implement basic web scraping for threat actors.

3. Implement basic web scraping for techniques.

4. Implement basic web scraping for malwares.

5. Save scraped data to CSV files.

6. Handle basic error logging and retries.

#### Sprint 2: Web Portal Development

1. Set up Flask for web portal.

2. Create a route to display threat actors.

3. Create a route to display techniques.

4. Create a route to display malwares.

5. Style the web portal using Bootstrap.

#### Sprint 3: Automation and Scheduling

1. Set up a task scheduler for automated scraping.

2. Test automated scraping process.

3. Implement notification system for scraping failures.

#### Sprint 4: Optional Enhancements (Database Integration)

1. Integrate database for data storage.

2. Modify scraping scripts to save data to database.

3. Modify Flask application to retrieve data from database.

4. Test the entire workflow from scraping to data display.

**### Mail Description**

Please find attachment for Mini-project details. Scrape the below urls and store the data in database like SQL, finally Prepare UI screens to display scraped data using REST APIs.

<https://attack.mitre.org/versions/v15/techniques/enterprise/>

<https://attack.mitre.org/versions/v15/groups/>

<https://attack.mitre.org/versions/v15/software/>

Use below example to scrape the data and store it in a specific format.

**Example**:

{

  "type": "Malware",

  "name": "3PARA RAT",

  "description": "3PARA RAT is a remote access tool (RAT) programmed in C++ that has been used by Putter Panda",

  "group": " Putter Panda",

  "techniques": [

    " File and Directory Discovery",

    "Indicator Removal: Timestomp"

  ]

}