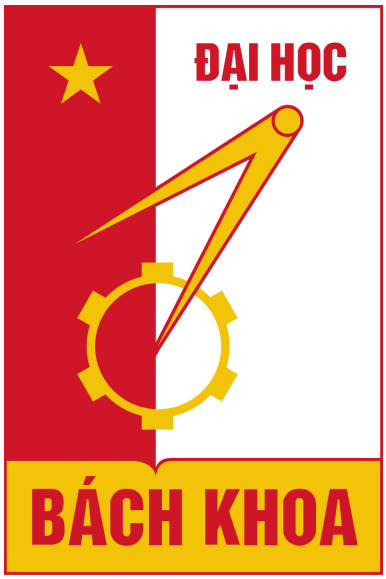
Hanoi University of Science and Technology
School of Information and Communication Technology
——————— o0o ———————



Atomic Red Team Excel Export and MITRE ATT&CK Coverage Analysis

Project I (IT3910E)

Supervisor : Tong Van Van

Students : Nguyen Quang Trung - 20225557
Nguyen Minh Duc 20225567

Hanoi - 2024

Contents

Abstract

This report documents the development of a Java-based application designed to enhance cyber security capabilities through the collection and analysis of threat intelligence from the open-source Atomic Red Team framework. The application's primary objectives include capturing and organizing data from Atomic Red Team simulations and exporting it to Excel for streamlined management and analysis. Additionally, the application provides coverage analysis of MITRE ATT&CK techniques, aiding cyber security professionals in evaluating and fortifying defensive strategies. The report details the methodology, implementation, functionality, and evaluation of the application, emphasizing its role in strengthening cyber security readiness in response to evolving threats.

Chapter 1

Introduction

In today's dynamic cyber-security landscape, organizations face continual challenges in defending against evolving threats. The Atomic Red Team, an open-source framework, provides valuable tools for simulating real-world cyber-attacks, while the MITRE ATT&CK Framework offers a structured approach to understanding and categorizing these attacks based on tactics and techniques. This report focuses on the development of

a Java-based application aimed at enhancing cybersecurity readiness through two primary objectives. Firstly, the application collects and organizes data from the Atomic Red Team repository, facilitating comprehensive insight into simulated cyber-attacks. Secondly, it integrates functionality to export this data to Excel files, streamlining data management and analysis. Additionally, the application includes features to analyze coverage across MITRE ATT&CK techniques, empowering cyber-security professionals to assess and enhance defensive strategies effectively.

The following sections detail the methodology, implementation, functionality, and evaluation of the Atomic Red Team Excel Export and MITRE ATT&CK Coverage Analysis application, highlighting its significance in bolstering organizational cyber-security defenses.

Chapter 2

Project Context

This section delves into the key components shaping the "Atomic Red Team Col lector" project's data collection and analysis. The MITRE ATT&CK Framework plays a vital role in modern cybersecurity defense, providing insights into adversary tactics and techniques. Additionally, the Atomic Red Team project serves as a valuable repository of adversary emulation plans, empowering cybersecurity teams with real-world cyber-attack simulations. While initially considering the GitHub API for data retrieval, the project took a different approach. This section provides a comprehensive exploration of these components, laying the groundwork for understanding the development and implementa tion of the "Atomic Red Team Collector" project.

2.1 Mitre Attack Framework

The MITRE ATT&CK Framework serves as a valuable knowledge base of adver sary tactics and techniques, derived from real-world observations. Its primary purpose is to provide a foundation for developing specific threat models and methodologies across various sectors, including the private sector, government, and the cybersecurity product and service community. MITRE's mission is to foster collaboration and enhance cyberse curity practices to create a safer world. ATT&CK is freely accessible to all, encouraging widespread use and contribution to the cybersecurity community. It offers a standard ized taxonomy for categorizing adversarial behaviors, aiding cybersecurity professionals in understanding adversary tactics and techniques, and effectively defending against cyber threats.

During the "Atomic Red Team Collector" project, we explored the MITRE ATT&CK website[4], where we found the "Working with ATT&CK" section. This section, accessible under the "Resources" menu, provided data access in two formats: Excel and STIX. While both formats were available, we chose to work with the STIX format due to its structured and machine-readable language, which facilitates the exchange of cyber threat intelligence and seamless integration with various cybersecurity tools, aligning with our project's objectives.

STIX, which stands for Structured Threat Information Expression, is a standardized language and serialization format used to exchange cyber threat intelligence. It enables the structured representation and sharing of cybersecurity data, including information about tactics, techniques, and procedures employed by adversaries.

To effectively work with the MITRE ATT&CK STIX data, we referred to the com prehensive documentation available in the STIX data repository[3]. This invaluable re source not only provided access to the STIX data itself, but also proved instrumental in understanding essential information, such as field names and value types. This under standing was necessary for parsing and utilizing the ATT&CK data seamlessly within our "Atomic Red Team Collector" application.

TheSTIX data for ATT&CK encompasses key fields crucial for our analysis, allowing us to gain valuable insights into cyber-attack techniques:

In addition to supporting coverage analysis, the STIX data for the technique provides valuable details about the technique itself, including its attributes, subtechniques, and related data. This information enhances the application's overall functionality, providing cybersecurity professionals with a comprehensive understanding of the characteristics and behaviors associated with each technique.

The Atomic Red Team project also provides STIX-formatted data, enabling easy in tegration of its valuable threat intelligence into our application. Further details about the Atomic Red Team project will be discussed in the following subsection.

2.2 Atomic Red Team

The Atomic Red Team project plays a critical role in providing adversary emulation plans for cybersecurity professionals by offering a comprehensive library of tests designed to simulate adversarial activities and validate defense mechanisms. These focused tests have minimal dependencies and are structured in a format compatible with automation frameworks, enabling efficient assessment of defense capabilities against real-world cyber threats.

The project's primary objective is to deliver high-quality adversary emulation plans that seamlessly integrate into security testing and assessment processes. By providing a vast collection of well-structured and user-friendly test cases, the Atomic Red Team empowers cybersecurity professionals to gain valuable insights into adversary behaviors and tactics, enhancing their understanding of potential attack vectors and strengthening their security posture.

Accessible on GitHub[1], the project's open-source repository contains an extensive collection of tests and techniques sourced from the reputable MITRE ATTCK Frame work. Unlike the MITRE ATTCK Framework's JSON format, the Atomic Red Team stores its data in YAML while adhering to the STIX format.

Each technique within the repository includes the essential atomic_test field, pro viding vital information on associated test cases, such as name, auto_generated_guid, de scription,

supported platform, input_arguments, executor, dependency_executor_name, and dependencies.

Additionally, the official Atomic Red Team website[2] offers valuable resources, in cluding coverage statistics illustrating technique coverage across various platforms. These statistics play a crucial role in assessing organizational preparedness against specific cyber threats and identifying potential security gaps.

In our project, the "Atomic Red Team Collector", we leverage the rich data available in the atomic$_{t}estfieldfromtheAtomicRedTeamrepository.Furthermore, toensuretheaccuracyandrelia$

2.3 Github API

In our project, all the necessary data is obtained from the official GitHub repositories of both the objects (MITRE ATT&CK[3] and Atomic Red Team[1]) that are in need of having their data collected. Therefore, we considered utilizing the GitHub API[9] in order to retrieve data from these two sources, specifically the JSON files for the MITRE ATT&CK and the YAML files for the Atomic Red Team.

After doing some study on the GitHub API, we came to the conclusion that this API cannot assist us in downloading files in a direct manner. Only visibility into the con tents of the repository is offered through the GitHub API. The information field labelled "download_url" will be included in the returned result if the item in query is a file. The "download_url" variable stores the URL that can be used to directly download the file. This URL looks like this:

raw.githubusercontent.com/$user/$repo/$branch/$path

Within those:

- user: the GitHub username of the user or organization that owns the repository.
- repo: the name of the GitHub repository you want to access.
- branch: the name of the branch from which you wish to retrieve the file.
- path: the relative path to the file within the repository. It specifies the location of the file you want to access.

Because of this, we are not making direct use of the GitHub API in this project; rather, we are making use of the download URL in order to get the necessary data files. As shown below:

- The file "atomics/Indexes/index.yaml" in the Atomic Red Team's repository[1] con tains all the data that is required for the atomic red team to function properly. Therefore, the part after raw.githubusercontent.com will look like: /redcanary co/atomic-red-team/master/atomics/Indexes/index.yaml

- MITRE ATT&CK divides Techniques into three categories based on their purpose: enterprise, mobile, and ics. is equivalent to the path "enterprise-attack/enterprise attack.json", "mobile-attack/mobile-attack.json", and "ics-attack/ics-attack.json" in the STIX data repository[3], so there will be three URLs as follows:
  - file:///mitre-attack/attack-stix-data/master/enterprise-attack/enterprise-attack.json
  - file:///mitre-attack/attack-stix-data/master/mobile-attack/mobile-attack.json
  - file:///mitre-attack/attack-stix-data/master/ics-attack/ics-attack.json

Chapter 3

Design

3.1 Use-case Diagram

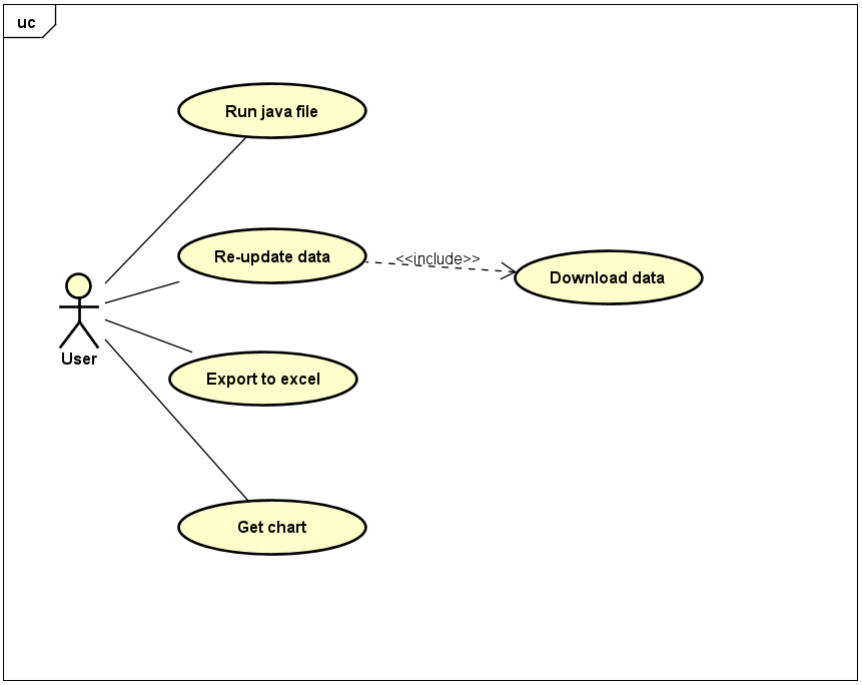


Figure 3.1: Use-case Diagram

This use case diagram outlines the primary interactions between users and the system for downloading updated data, retrieving charts, and exporting data to Excel. Each function is designed to enhance usability and efficiency within the system.

Users

- **User**: Represents users utilizing the system.

Functions

1. Re-update Data (Download Data)
    - **Description**: Allows users to download updated data
    - **Actors**: User
    - **Preconditions**: User is authenticated and has access rights
    - **Flow of Events:**
        - User initiates the re-update data process.
        - System retrieves and downloads the latest data.
        - System confirms successful download to the user
    - **Post conditions**: Updated data is available to the user
2. Get Chart
    - **Description**: Allows users to download updated data
    - **Actors**: User
    - **Preconditions**: User is authenticated and has access rights
    - **Flow of Events:**
        - User requests a specific chart.
        - System generates the requested chart.
        - System displays the chart to the user
    - **Post conditions**: User views the requested chart.
3. Export Excel
    - **Description**: Allows users to download updated data
    - **Actors**: User
    - **Preconditions**: User is authenticated and has access rights
    - **Flow of Events:**
        - User request an excel.
        - System generates an excel.
        - System displays the excel to the user
    - **Post conditions**: User downloads the exported data in Excel format.

3.2 Class Diagram

3.2.1 General Class Diagram

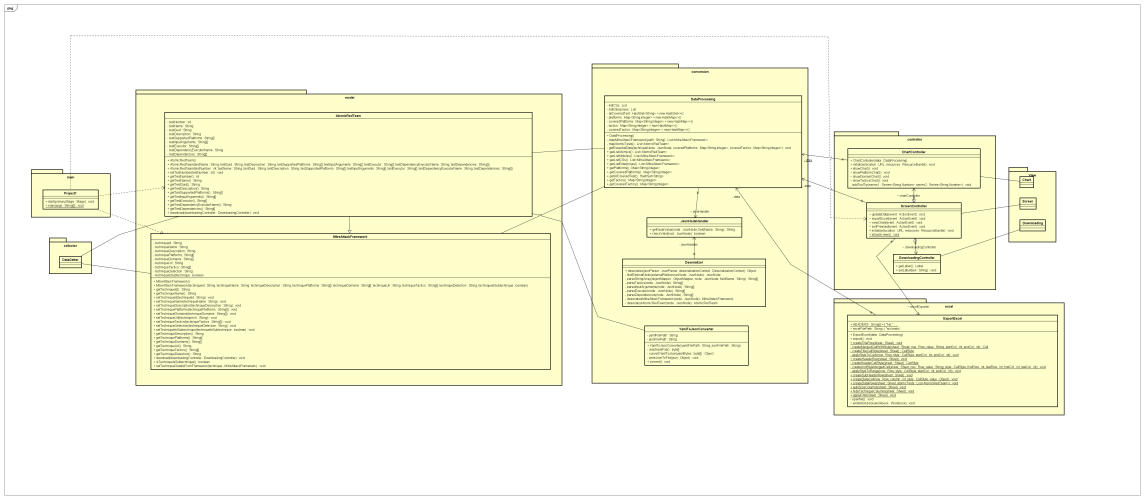


Figure 3.2: General Class Diagram

This class diagram provides an overview of the system's architecture, showing how different classes are structured and how to interact with each other.

3.2.2 dataGetter Class

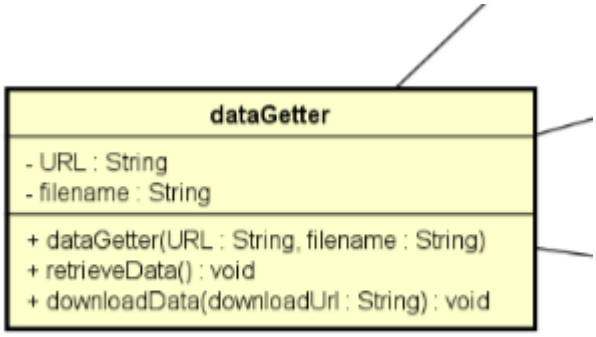


Figure 3.3: dataGetter class

Overview

The `dataGetter` class is a Java class designed to retrieve data from a specified URL and save it to a file. It uses the Java HTTP Client API for making HTTP requests and

handling responses.

Fields

- `private final String URL;`
  - The URL from which data is to be retrieved.
- `private final String filename;`
  - The name of the file to which the data will be saved.
- `private long startTime, endTime, elapsedTime;`
  - Variables to track the time taken to retrieve and save the data.

Constructor

Initializes the `URL` and `filename` fields with the given values.

Methods

`public void retrieveData(boolean atomic) throws IOException` This method is responsible for making an HTTP GET request to the specified URL, processing the response, and calling the `downloadData` method to download the actual data from the download URL obtained from the response.

Steps

1. **Timing the Operation**
2. **Building the HTTP Request**
3. **Creating the HTTP Client**
4. **Sending the Request and Handling the Response**

`public void downloadData(String downloadUrl, boolean atomic) throws IOException` This method downloads the data from the given `downloadUrl` and saves it to a file in a specified directory.

Steps

1. **Determining the Directory Path**
2. **Ensuring the Directory Path Ends with a Separator**
3. **Downloading and Saving the Data**

`public String getElapsedTime()` This method returns the elapsed time in seconds as a string.

3.2.3 Model Package

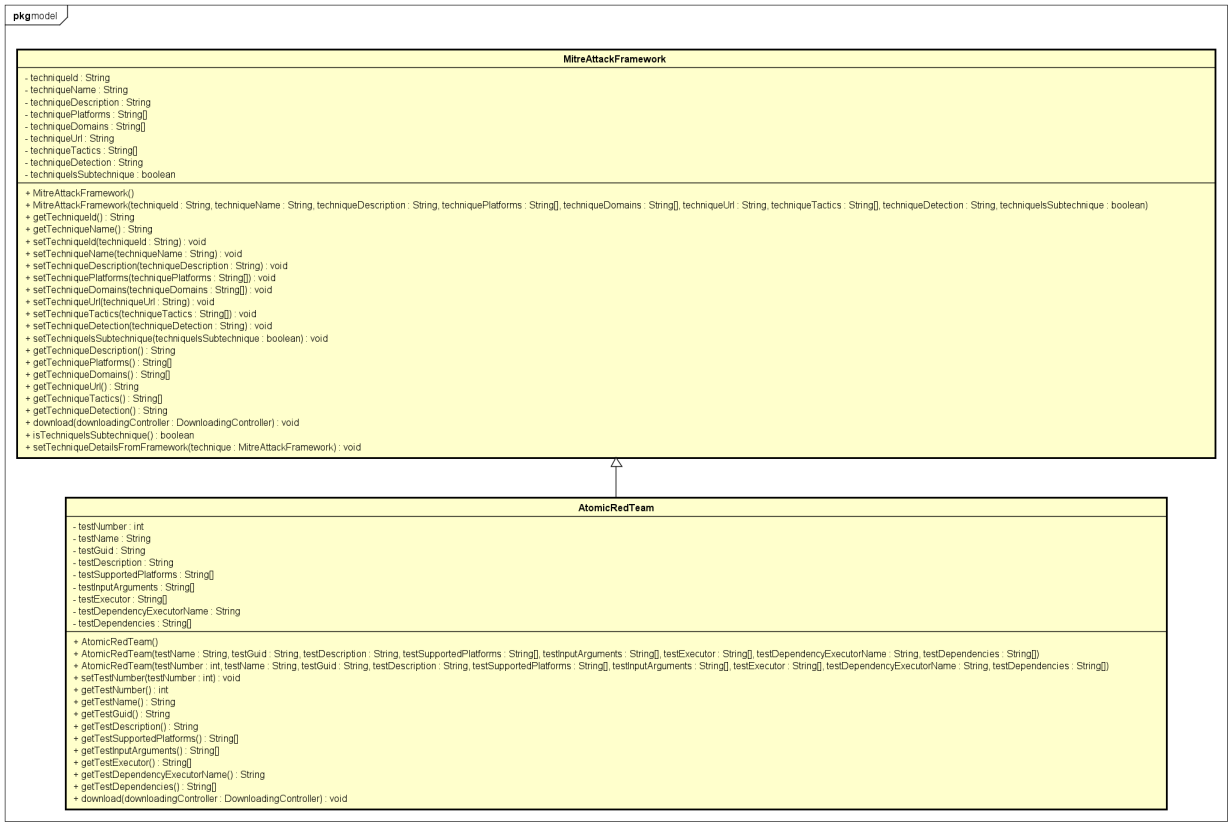


Figure 3.4: Model Diagram

AtomicRedTeam class:

- **Attributes**
  - **testNumber**: an integer representing the test number.
  - **testName, testGuid, testDescription**: Strings for the test's name, globally unique identifier, and description respectively. These are annotated with @JsonProperty for JSON serialization.
  - **testSupportedPlatforms**, **testInputArguments**, **testExecutor**, **testDependencies**: Arrays of strings representing the platforms supported by the test, input arguments for the test, executor details, and any dependencies.

  - **testDependencyExecutorName**: a string for the name of the dependency executor.

- **Method**

  1. **Initialization:**
     - ATOMIC_URL and NAME_FILE are defined for the URL of the YAML file and its local name.
     - dataGetter object (atomicRetriever) is created to handle the download.
  2. **Download Process:**
     - The label on the downloadingController is updated to indicate the download start.
     - The atomicRetriever attempts to retrieve the data, printing any exceptions.
     - After the download, the label is updated with the elapsed time of the download.
  3. **Conversion:**
     - Paths for the YAML and JSON files are defined.
     - A YamlToJsonConverter object is created to convert the downloaded YAML file to JSON.
     - The label is updated to indicate the conversion process, and the conversion is performed.
     - Finally, the label is updated to indicate the completion of the conversion.

MitreAttackFramework class:

Package and Imports

- The class is part of the `hust.cybersec.model` package.
- Various classes and annotations are imported from libraries such as Jackson for JSON processing, custom classes, and JavaFX for UI updates.

Class Definition and Annotations

- The class represents an entity in the MITRE ATT&CK framework.
- `@JsonDeserialize`: Specifies a custom deserializer for JSON data.
- `@JsonIgnoreProperties`: Ignores any unknown properties during JSON deserialization.

Fields

- `techniqueId`: Stores the ID of the technique.
- `techniqueName`: Stores the name of the technique.
- `techniqueDescription`: Stores the description of the technique.
- `techniquePlatforms`: Stores the platforms associated with the technique.
- `techniqueDomains`: Stores the domains associated with the technique.
- `techniqueUrl`: Stores the URL of the technique.
- `techniqueTactics`: Stores the tactics associated with the technique.
- `techniqueDetection`: Stores detection information for the technique.
- `techniqueIsSubtechnique`: Indicates if the technique is a sub-technique.

Constructors

- `MitreAttackFramework()`: Initializes an empty `MitreAttackFramework` object.
- `MitreAttackFramework(String, String, String, String[], String[], String, String[], String, boolean)`: Initializes a `MitreAttackFramework` object with specified values for all fields.

Getters and Setters

- **Getters:** Methods to retrieve the values of the fields.
- **Setters:** Methods to set the values of the fields.

Methods

download

- **Description:** Downloads data from specified URLs and updates the downloading status.
- **Parameters:**
  - `DownloadingController downloadingController`: Controller for handling download status updates.
- **Exception:** `URISyntaxException` may be thrown.

- **Details:**
  - **MITRE_URL:** Array of URLs to download JSON data from MITRE ATT&CK.
  - **NAME_FILE:** Corresponding filenames for the downloaded data.
  - **Loop:** Iterates through the URLs to download data.
  - **dataGetter:** Creates a `dataGetter` object for each URL.
  - **Platform.runLater:** Updates UI labels to indicate downloading status.
  - **Exception Handling:** Catches and logs errors during the download process.

setTechniqueDetailsFromFramework

- **Description:** Copies details from another `MitreAttackFramework` object to the current object.
- **Parameter:** `MitreAttackFramework technique` - The source object from which details are copied.
- **Details:** Calls setters to update the fields of the current object with values from the provided object.

3.2.4 Conversion package

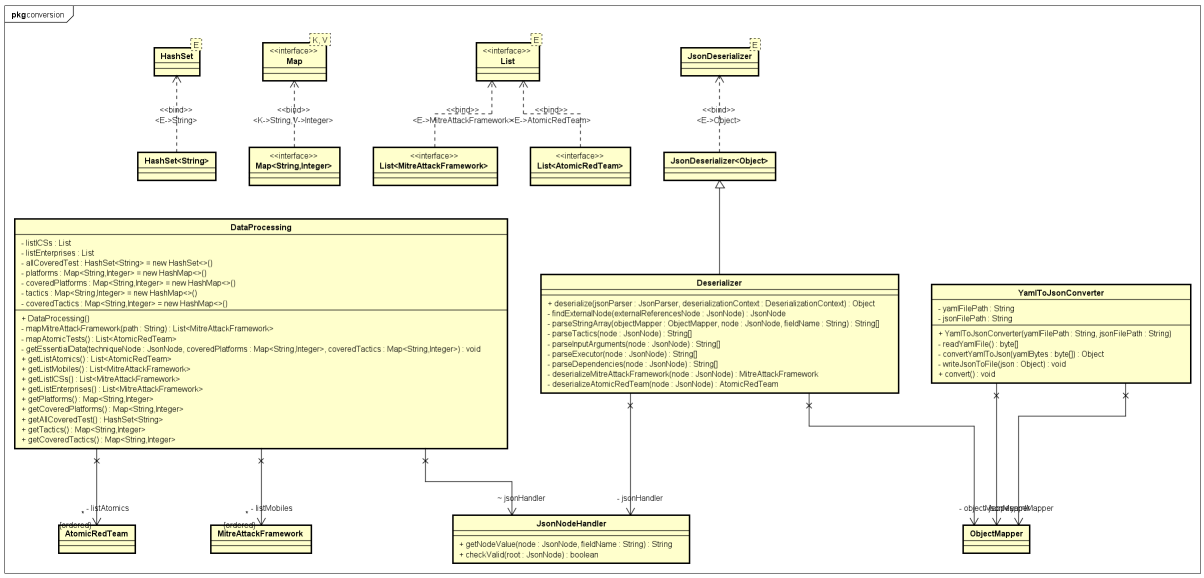


Figure 3.5: Model Diagram

DataProcessing class

Package and Imports

- The class is part of the `hust.cybersec.conversion` package.
- Imports necessary libraries and classes for JSON processing, file operations, and collections handling.

Class Definition and Fields

- `listAtomics`, `listMobiles`, `listICSs`, `listEnterprises`: Lists to store instances of `AtomicRedTeam` and `MitreAttackFramework`.
- `jsonHandler`: Instance of `JsonNodeHandler` for JSON node processing.
- Maps (`platforms`, `coveredPlatforms`, `tactics`, `coveredTactics`): Store platform and tactic data.

Constructor

- Initializes lists (`listAtomics`, `listMobiles`, `listICSs`, `listEnterprises`) by mapping JSON files.

Methods

mapMitreAttackFramework

- Reads and maps JSON data from the specified `path` into `MitreAttackFramework` objects.

mapAtomicTests

- Reads and maps JSON data from the Atomic Red Team file into `AtomicRedTeam` objects.

getEssentialData

- Extracts essential data (platforms and tactics) from a given JSON node.

Getters

- Get methods for retrieving various data structures (`listAtomics`, `listMobiles`, etc.) and maps (`platforms`, `coveredPlatforms`, etc.).

Deserializer class

Package and Imports

- The class is part of the `hust.cybersec.conversion` package.
- Imports necessary Jackson libraries and classes for JSON processing and handling.

Class Definition and Fields

- `Deserializer` extends `JsonDeserializer<Object>`, indicating it can deserialize JSON into Java objects.
- `objectMapper`: Instance for JSON parsing and mapping.
- `jsonHandler`: Instance for handling JSON nodes.

Methods

deserialize

- Entry point for deserialization. Determines the type of object(`AtomicRedTeam` or `MitreAttackFramew` based on the presence of $\texttt{auto}_g enerated_g uid in the JSON node$.

Listing 3.1: findExternalNode Method

```
private JsonNode findExternalNode(JsonNode externalReferencesNode) {
    // Method implementation
}
```

- Searches for and returns the external node related to MITRE ATTCK framework within the $\texttt{external}_r eferences JSON node$.

parseStringArray

- Parses a JSON array into a String array for the specified field name.

parseTactics, parseInputArguments, parseExecutor, parseDependencies

- Various utility methods to parse specific JSON structures (tactics, input arguments, executor details, dependencies) into arrays of strings.

deserializeMitreAttackFramework

- Deserializes a JSON node into a `MitreAttackFramework` object, extracting relevant fields like ID, name, description, platforms, domains, URL, tactics, detection, and subtechnique status.

deserializeAtomicRedTeam

- Deserializes a JSON node into an `AtomicRedTeam` object, extracting fields related to test name, GUID, description, supported platforms, input arguments, executor details, dependency executor name, and dependencies.

JsonNodeHandler

Package and Imports

- The class is part of the `hust.cybersec.conversion` package.
- Imports `JsonNode` from Jackson library for JSON handling.

Class Definition and Methods

- The `JsonNodeHandler` class provides utility methods to handle JSON nodes.
- `getNodeValue`: Retrieves the text value of a specified field from a JSON node. Returns an empty string if the field does not exist.
- `checkValid`: Checks if a JSON node represents a valid attack pattern from MITRE ATTCK based on specific criteria:
  - The node must have a type of "attack-pattern".
  - It must not be marked as revoked or deprecated.
  - It must have external references, and at least one reference should be from "mitre-attack" with an ID starting with "T".

YamlToJsonConverter

Package and Imports

- The class is part of the `hust.cybersec.conversion` package.
- Imports necessary classes for file handling, YAML and JSON processing from Jackson library, and LoaderOptions from SnakeYAML.

Class Definition and Constructor

- The `YamlToJsonConverter` class is designed to convert YAML files to JSON format.
- It initializes with paths to input (`yamlFilePath`) and output (`jsonFilePath`) files.
- Configures a YAML factory (`yamlFactory`) with custom options to handle larger files.
- Creates instances of `ObjectMapper` for YAML (`yamlMapper`) and JSON (`jsonMapper`).

Utility Methods

- `readYamlFile()`: Reads the YAML file specified by `yamlFilePath` and returns its content as a byte array.
- `convertYamlToJson(byte[] yamlBytes)`: Converts the provided YAML byte array to a Java object using `yamlMapper`.
- `writeJsonToFile(Object json)`: Writes the given Java object (`json`) as JSON to the file specified by `jsonFilePath` using `jsonMapper`.

Conversion Method

- `convert()`: Method that orchestrates the entire YAML to JSON conversion process.
- It reads the YAML file, converts it to a Java object, and then writes the JSON representation to the output file.
- Handles potential `IOExceptions` during file operations.

3.2.5 ExportExcel class

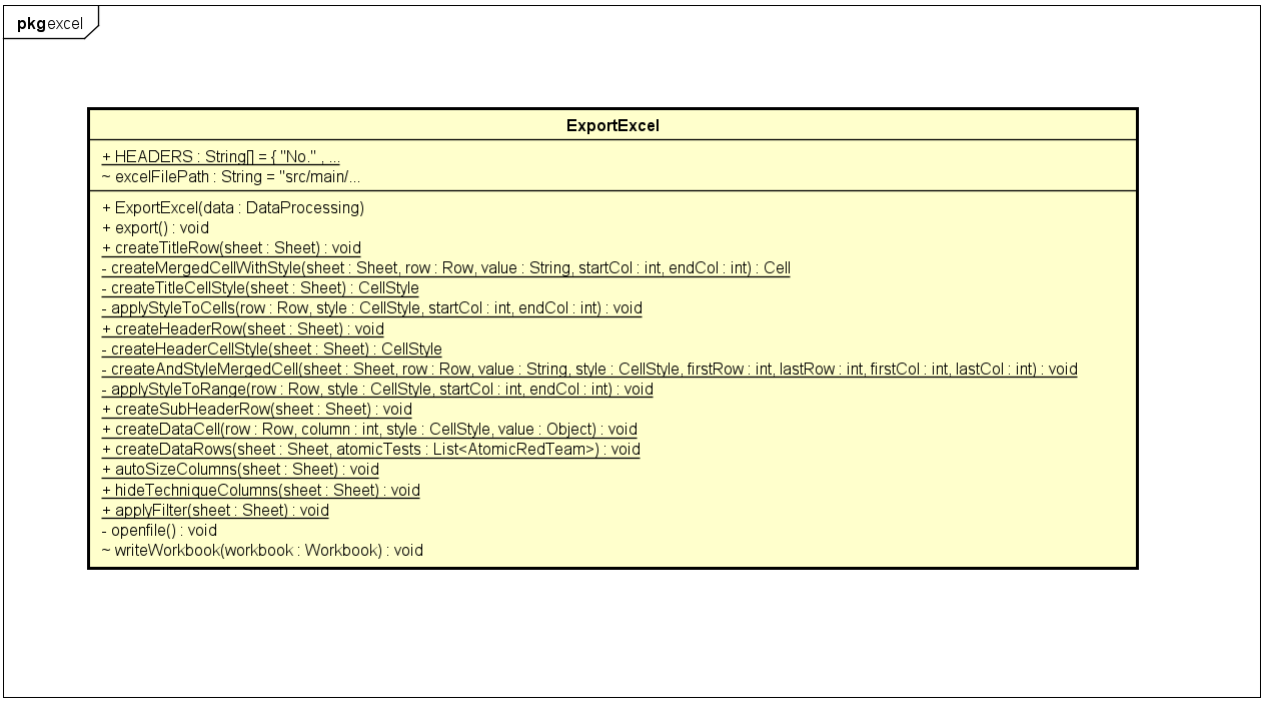


Figure 3.6: ExportExcel class Diagram

Overview

The ExportExcel class is designed to export data related to Atomic Red Team tests into an Excel spreadsheet format (.xlsx). It utilizes Apache POI, a popular Java library for working with Microsoft Office documents.

Class Structure

- **Package and Imports**: The class is located in the `hust.cybersec.excel` package and imports necessary classes from Apache POI and other Java libraries for file handling and data processing.
- **Constants**: Defines HEADERS, an array containing column headers for the Excel spreadsheet.
- **Instance Variables**:
  - excelFilePath: Specifies the file path where the Excel file will be created.
  - data: An instance of DataProcessing, which presumably provides the data to be exported.
- **Constructor**: Initializes the data object using the DataProcessing class.
- **Methods**:
  - export(): Orchestrates the entire process of Excel export. Initializes a new Workbook instance (XSSFWorkbook), creates a sheet named "Atomic Test", and sets a zoom level for better visibility.
  - **Data Layout**:
    - createTitleRow(sheet): Merges cells and creates a title row with a specific style.
    - createHeaderRow(sheet): Sets up header rows ("TECHNIQUE" and "TEST") and sub-header rows using merged cells and predefined styles.
    - createDataRows(sheet, data.getListAtomics()): Populates the Excel sheet with data rows based on the provided AtomicRedTeam objects.
  - **Utility Methods**: Includes methods for creating cells, applying styles, auto-sizing columns, hiding specific columns, applying filters, writing the workbook to a file, and opening the generated Excel file.

Usage

The export() method is the entry point for exporting data to Excel. After the export operation, it prints a success message and attempts to open the generated Excel file using the openfile() method.

Dependencies

- Relies on Apache POI for Excel-related functionalities (Workbook, Sheet, Row, Cell, etc.).
- Uses DataProcessing for retrieving and processing data to be exported.

3.2.6 Controller class Diagram

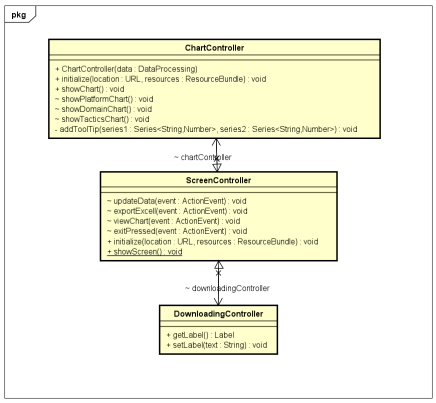


Figure 3.7: Controller Diagram

ScreenController class

Overview

The ScreenController class is a controller in a JavaFX application responsible for managing different screens and user interactions. It initializes necessary components, handles button actions to update data, export data to Excel, view charts, and exit the application.

Class Structure

- **Package and Imports**: The class is located in the `hust.cybersec.controller` package and imports necessary JavaFX, I/O, and model classes (DataProcessing, ExportExcel, AtomicRedTeam, MitreAttackFramework, etc.).
- **Instance Variables**:
  - data: An instance of DataProcessing used for processing and managing data.
  - excelExporter: An instance of ExportExcel used for exporting data to Excel format.
  - downloadingController, chartController: Controllers for managing downloading progress and displaying charts respectively.

  - maf, art: Instances of MitreAttackFramework and AtomicRedTeam for handling data related to these frameworks.

- **FXML Annotations and Methods**:
  - updateData(ActionEvent event): Handles the button action to update data. It initiates a download task using JavaFX concurrent API, updates UI on success or failure, and transitions to a downloading screen.
  - exportExcel(ActionEvent event): Handles the button action to export data to Excel using the excelExporter instance.
  - viewChart(ActionEvent event): Handles the button action to view charts, calling showChart() on the chartController instance.
  - exitPressed(ActionEvent event): Handles the button action to exit the application using Platform.exit().
  - initialize(URL location, ResourceBundle resources): Initializes the controller, sets up instances of data, excelExporter, downloadingController, maf, and art.

- **Utility Methods**:
  - showScreen(): Static method to show the main screen defined in `/View/Screen.fxml`. It sets up the scene and stage for the primary window of the application.

Usage

The ScreenController class serves as the main controller for managing screens and user interactions in the application. It handles data updates, Excel export, chart viewing, and application exit.

Dependencies

- Utilizes JavaFX for building the user interface.
- Relies on DataProcessing, ExportExcel, AtomicRedTeam, and MitreAttackFramework for data processing, exporting, and framework-specific operations.

Downloading Controller

Overview

The DownloadingController class is a specialized controller in a JavaFX application responsible for managing the downloading screen. It extends from ScreenController and implements Initializable to initialize JavaFX components.

Class Structure

- **Package and Imports**: The class is located in the `hust.cybersec.controller` package and imports necessary JavaFX classes and interfaces (FXML, FXMLLoader, Label, etc.).
- **FXML Annotations and Fields**:
  - label: An instance variable annotated with @FXML to reference a Label component defined in the corresponding FXML file.
- **Getter and Setter Methods**:
  - getLabel(): Returns the Label instance associated with the label field.
  - setLabel(String text): Sets the text of the Label instance. It appends the provided text to the existing text in the label.

Usage

The DownloadingController class is used to control the downloading screen in the application. It provides methods to update the status label dynamically during the download process.

Inheritance

- Inherits from ScreenController: Extending functionalities defined in ScreenController allows DownloadingController to reuse common UI handling logic, such as transitioning between screens.

Dependencies

- Relies on JavaFX for building the user interface and managing UI components.
- Extends functionality from ScreenController, inheriting behavior for managing screens and user interactions.

ChartController class

Overview

The ChartController class in the `hust.cybersec.controller` package is a JavaFX controller responsible for managing charts in an application related to cyber security analysis. It extends from ScreenController and implements Initializable to initialize JavaFX components.

Class Structure

- **Imports**: Imports necessary JavaFX and other classes (FXML, FXMLLoader, StackedBarChart, ChoiceBox, etc.).
- **Fields**:
  - data: A static field of type DataProcessing to store data for chart visualization.
  - stackedChart: Annotated with @FXML, it represents a StackedBarChart component defined in the corresponding FXML file.
  - choiceBox: Annotated with @FXML, it represents a ChoiceBox¡String¿ component for selecting chart categories.
- **Constructor**:
  - ChartController(DataProcessing data): Initializes the data field with the provided DataProcessing instance.
- **Initialization**:
  - initialize(URL location, ResourceBundle resources): Initializes the ChoiceBox with chart categories ("Domains", "Platforms", "Tactics") and sets listeners to display corresponding charts based on user selection. Additionally, sets the title and animation for the stackedChart.
- **Chart Display Methods**:
  - showChart(): Loads and displays the chart scene using FXMLLoader with the provided data.
  - showPlatformChart(), showDomainChart(), showTacticsChart(): Methods to populate and display specific types of charts (StackedBarChart) based on the selected category in the choiceBox.
- **Utility Methods**:
  - addToolTip(XYChart.Series¡String, Number¿ series1, XYChart.Series¡String, Number¿ series2): Adds tooltips to chart data points showing the value and coverage ratio.

Usage

The ChartController class is used to dynamically display and manage different types of stacked bar charts representing cyber security analysis data. It leverages JavaFX for UI components and chart visualization.

Dependencies

- Utilizes JavaFX for building and managing the user interface, including chart components (StackedBarChart, ChoiceBox).
- Depends on DataProcessing for accessing and processing underlying data related to cyber security analysis.

Chapter 4

Demonstration and User Guide

4.1 Demonstration

The main java class of this project is Project1.java in **main** folder

When you start running the project, it will show you this:

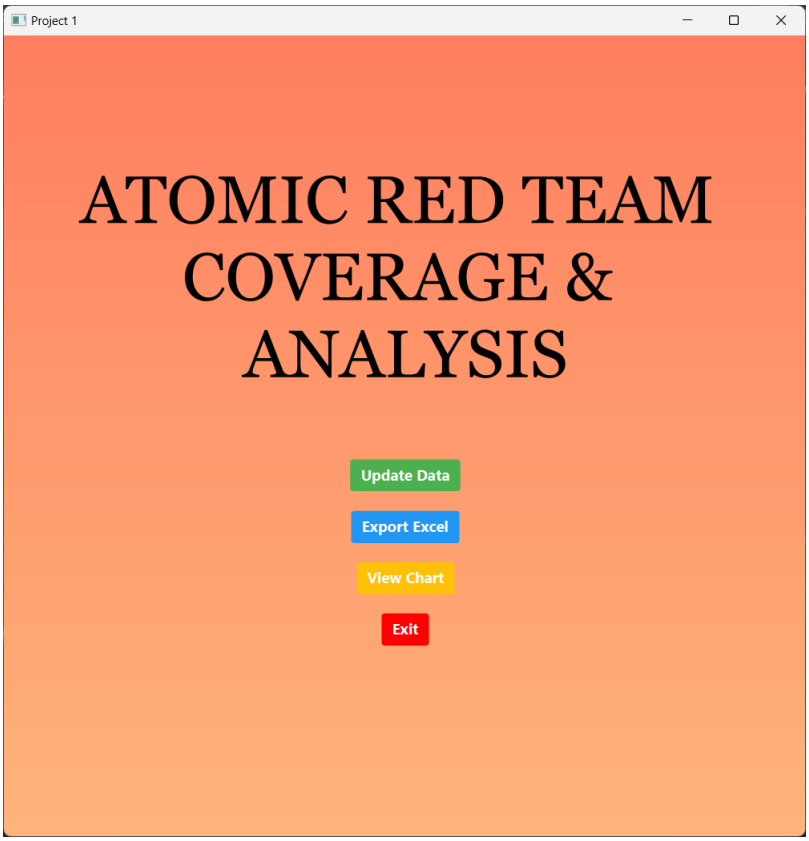


Figure 4.1: Main UI

This is the main UI for this project. As you can see, it has four buttons for four functions.

When you start pressing the "Update Data" button, it will shows this panel:

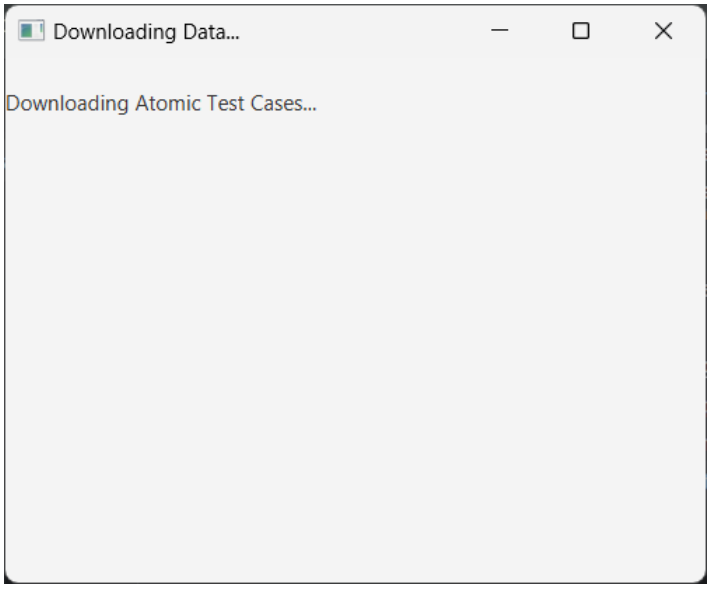


Figure 4.2: Downloading UI

This is the log panel to show the user how much data have downloaded.
And when it's done downloading, it will show again the main panel of this project.
And again, when you start pressing the "Export Excel" button, it will shows this panel:

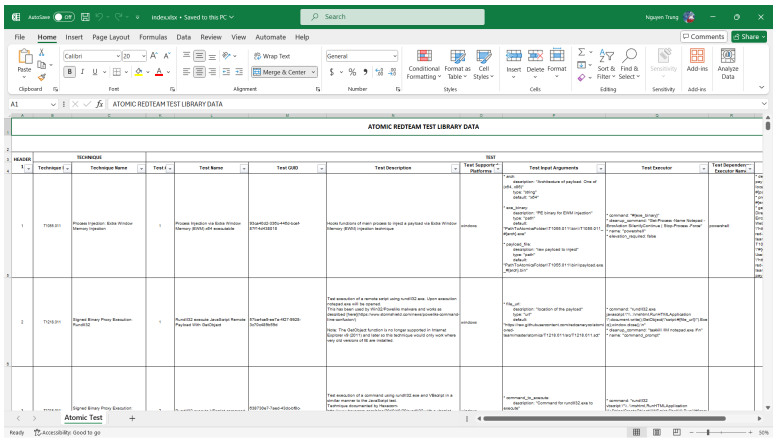


Figure 4.3: Excel

This is the excel that show all technique (in enterprise-attack) that Atomic Red Team has tested or not.

And again, when you start pressing the "Show Chart" button, it will shows this panel:

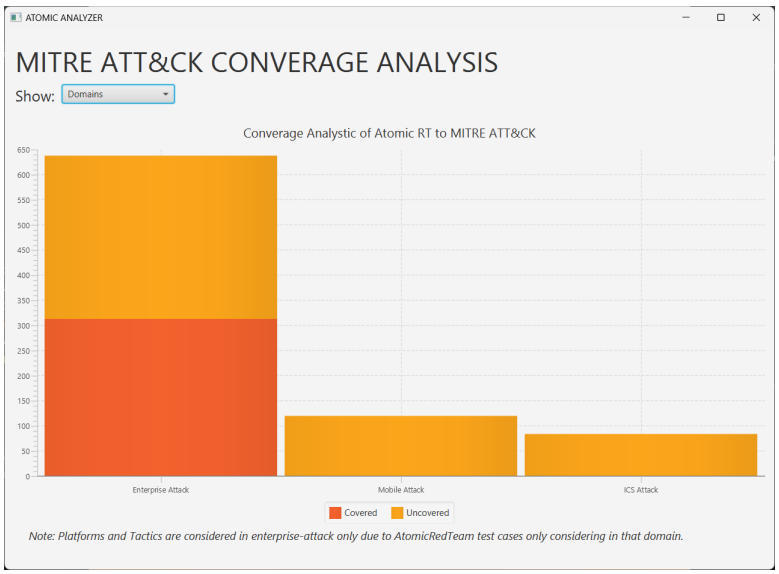


Figure 4.4: Chart UI

This is the chart that show how much Atomic Red Team has covered in Mitre Attack Framework

It also have 3 domain to compare AtomicRedTeam with MitreAttackFramework:

- Domain (compare with all 3 attack): enterprise-attack, mobile-attack and ics-attack.
- Platform (compare with enterprise-attack): show how AtomicRedTeam cover every platform on enterprise-attack.
- Tactics (compare with enterprise-attack): show how AtomicRedTeam cover every tactics on enterprise-attack.

4.2 User Guide

This can probably run on every Java IDE. The **main** file of this project is ProjectI.java class. If you are using JavaSDK version 8.x, run that file and you're ready to go.

If not, you should install javaFX SDK on the project.

4.2.1 On Eclipse

Install plugin for JavaFX

To work with JavaFX in Eclipse, you need to install some plugins. The most popular one is **e(fx)clipse**.

- Open Eclipse, on the Menu bar, choose Help → Eclipse MarketPlace → search e(fx)clipse and click Install.
- Follow the instructions and restart Workspace.
- Check after installation: After successful installation and restarting Eclipse, you can check the result of the installation. In Eclipse select: File/New/Others... There are Wizards which allow you to carry out JavaFX programming

Download JavaFX

Go to this page https://gluonhq.com/products/javafx/, select the latest JavaFX version then select your Operating System and Architecture, for the Type option select SDK. Download the JavaFX and extract it to a folder (please remember the location).

Add JavaFX to Eclipse

In this part, you will create a user library for JavaFX in Eclipse.

- Open Eclipse, on the Menu bar, choose Window → Preferences → search User Libraries → New → Name it as "JavaFX"
- Click "Add External JARs" then navigate to the folder where you extracted JavaFX in the previous step, choose the "lib" folder and add all .jar files. Click "Apply and Close".
- Right-click on the project → Build Path → Configure Build Path → Classpath → Add Library → User Library → JavaFX

Configuring Build Path and Arguments

You need to set up the run configuration by following these steps:

- Right-click on the project → Run As → Run Configurations → Arguments → VM arguments
- Add the following command: –module-path "YOUR/PATH/lib" –add-modules javafx.controls,ja
- Click Apply.

4.2.2 On IntelliJ

Download JavaFX

Go to this page https://gluonhq.com/products/javafx/, select the latest JavaFX version then select your Operating System and Architecture, for the Type option select SDK. Download the JavaFX and extract it to a folder (please remember the location).

Configuring Build Path and Arguments

You need to set up the run configuration by following these steps:

- Click on the Current File on top of the window → Edit Configuration .
- Click Add new configuration on the left.
- in the Main Class section should be: hust.cybersec.main.Project1
- Click the Modify options → Add VM Options.
- Add the following command in the VM Options: –module-path "YOUR/PATH/lib" –add-modules javafx.controls,javafx.fxml
- Click Apply → Run.

Chapter 5

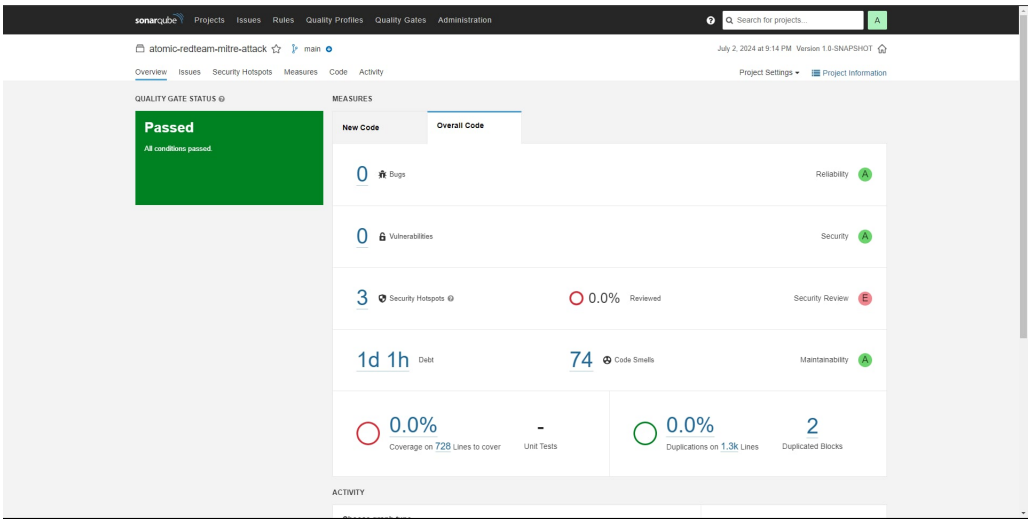
Code Quality Analysis

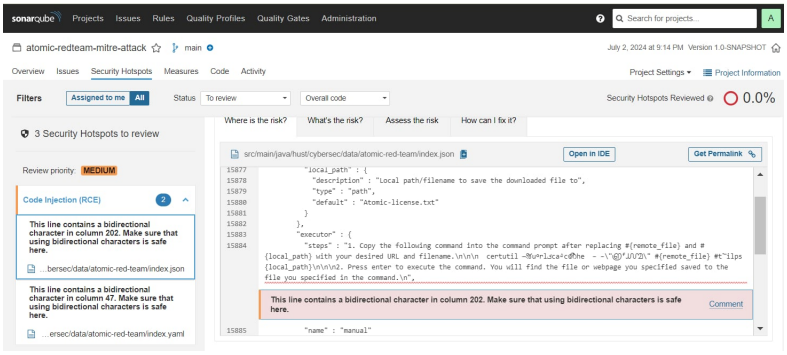
We implemented Jenkins for our program. Jenkins pipeline was set up to automates the build, test and deployment:

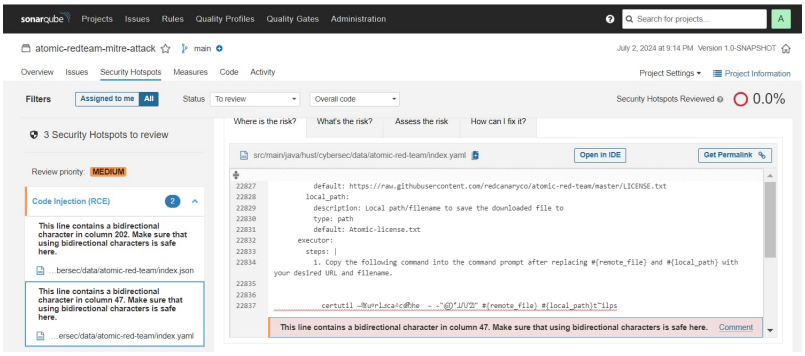
- **Build**: Jenkins automatically built the project upon new code commits, ensuring the latest code was complied and ready for testing
- **Test**: A suit of automated tests ran after each build to verify the code's functionality.
- **Deploy**: Jenkins deployed the tested code to a staging environment for further validation.

To maintain high code quality, we integrated SonarQube with Jenkins. SonarQube analyzed our code for potential bugs or vulnerabilities and provided detailed reports on code quality. Using SonarQube Scanner tool, SonarQube was integrated into Jenkins pipeline, which executed code analysis during Jenkins build process. The analysis included metrics such as bugs, vulnerabilities, code smells. SonarQube also displayed a real-time insights into code quality, highlighting areas that needed to be improved or to be fixed.

This is our result on Jenkins:


sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
Search for projects...
atomic-redteam-mitre-attack
main
July 2, 2024 at 9:14 PM Version 1.0-SNAPSHOT
Overview
Issues
Security Hotspots
Measures
Code
Activity
Project Settings
Project Information
QUALITY GATE STATUS
Passed
All conditions passed.
MEASURES
New Code
Overall Code
0 Bugs
Reliability A
0 Vulnerabilities
Security A
3 Security Hotspots
0.0% Reviewed
Security Review E
1d 1h Debt
74 Code Smells
Maintainability A
0.0%
Coverage on 728 Lines to cover
-
Unit Tests
0.0%
Duplications on 1.3k Lines
2
Duplicated Blocks
ACTIVITY


sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
Search for projects...
atomic-redteam-mitre-attack
main
July 2, 2024 at 9:14 PM Version 1.0-SNAPSHOT
Overview
Issues
Security Hotspots
Measures
Code
Activity
Project Settings
Project Information
Filters
Assigned to me
All
Status
To review
Overall code
Security Hotspots Reviewed
0.0%
3 Security Hotspots to review
Where is the risk?
What's the risk?
Assess the risk
How can I fix it?
Review priority MEDIUM
Code Injection (RCE) 2
This line contains a bidirectional character in column 202. Make sure that using bidirectional characters is safe here.
...bersec/data/atomic-red-team/index.json
This line contains a bidirectional character in column 47. Make sure that using bidirectional characters is safe here.
...ersec/data/atomic-red-team/index.yaml
src/main/java/hust/cybersec/data/atomic-red-team/index.json
Open in IDE
Get Permalink
"local_path" : {
"description" : "Local path/filename to save the downloaded file to",
"type" : "path",
"default" : "Atomic-license.txt"
}
},
"executor" : {
"steps" : "1. Copy the following command into the command prompt after replacing #{remote_file} and #{local_path} with your desired URL and filename.\n\n\n certutil -%u+rlca+cd%he - -\"@J*.l/\?2\" #{remote_file} #t^lips {local_path}\n\n\n2. Press enter to execute the command. You will find the file or webpage you specified saved to the file you specified in the command.\n",
This line contains a bidirectional character in column 202. Make sure that using bidirectional characters is safe here.
Comment
"name" : "manual"


sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
Search for projects...
atomic-redteam-mitre-attack
main
July 2, 2024 at 9:14 PM Version 1.0-SNAPSHOT
Overview
Issues
Security Hotspots
Measures
Code
Activity
Project Settings
Project Information
Filters
Assigned to me
All
Status
To review
Overall code
Security Hotspots Reviewed
0.0%
3 Security Hotspots to review
Where is the risk?
What's the risk?
Assess the risk
How can I fix it?
Review priority MEDIUM
Code Injection (RCE) 2
This line contains a bidirectional character in column 202. Make sure that using bidirectional characters is safe here.
...bersec/data/atomic-red-team/index.json
This line contains a bidirectional character in column 47. Make sure that using bidirectional characters is safe here.
...ersec/data/atomic-red-team/index.yaml
src/main/java/hust/cybersec/data/atomic-red-team/index.yaml
Open in IDE
Get Permalink
default: https://raw.githubusercontent.com/redcanaryco/atomic-red-team/master/LICENSE.txt
local_path:
description: Local path/filename to save the downloaded file to
type: path
default: Atomic-license.txt
executor:
steps: |
1. Copy the following command into the command prompt after replacing #{remote_file} and #{local_path} with your desired URL and filename.
certutil -%u+rlca+cd%he - -"@J*.l/\?2" #{remote_file} #{local_path}t^lips
This line contains a bidirectional character in column 47. Make sure that using bidirectional characters is safe here.
Comment

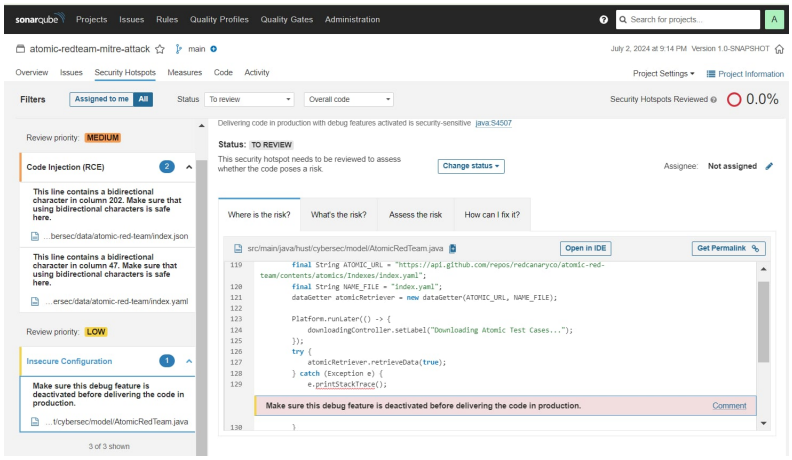


Figure 5.1: Results

There are three errors identified in the ***Security Hotspots*** section. The first two errors are located within the Atomic Red Team files, specially in the index.yaml and index.json files.

Chapter 6

Conclusion

Our project has successfully met its objectives, providing valuable insights to cybersecurity professionals through comprehensive data analysis, chart generation, and data storage. The implementation of data retrieval, download, and export functionalities ensures the accuracy and efficiency of the cyberattack database, empowering professionals with essential information for analysis and validation.

The chart analysis feature, in particular, excels in delivering insightful visualizations of technique coverage across various tactics, platforms, and domains. This aids cybersecurity professionals in identifying areas that require attention when creating test cases to strengthen their security defenses. Although the chart analysis results may differ from those on the official Atomic Red Team website due to differing approaches, the project's analysis remains valid and valuable. The number of test cases accurately reflects data from the Atomic Red Team's official GitHub repository and the total number of techniques provided by the MITRE ATT&CK Framework official website, ensuring relevance and reliability.

However, the project has some limitations. The current version's chart analysis feature is restricted to a single launch due to JavaFX framework limitations, and there is a lack of robust error handling during data retrieval. To enhance the overall user experience, future releases will prioritize implementing a graphical user interface (GUI) for the entire application and improving error handling mechanisms.

As the project continues to evolve, the current tree structure for storing collected data may become limiting. Future enhancements should consider storing data in a database, such as SQL, for more efficient and scalable data management. This approach will support the long-term growth and sustainability of the project.

Chapter 7

References

1 Red Canary. Atomic Red Team GitHub Repo. https://github.com/redcanaryco/atomic-red-team.

2 Red Canary. Explore Atomic Red Team. https://atomicredteam.io/

3 The MITRE Corporation. ATT&CK STIX Data GitHub Repo. https://github.com/mitre-attack/attack-stix-data

4 The MITRE Corporation. MITRE ATT&CK. https://attack.mitre.org/

5 FasterXML. FasterXML integrates SnakeYAML GitHub Repo. https://github.com/FasterXML/jac dataformats-text/tree/2.15/yaml

6 FasterXML. Jackson GitHub Repo. https://github.com/FasterXML/jackson

7 CD Foundation. Jenkins. https://www.jenkins.io/

8 The Apache Software Foundation. Apache POI. https://poi.apache.org/.

9 GitHub. GitHub REST API. https://docs.github.com/rest.

10 OpenJFX. JavaFX. https://openjfx.io.

11 Our Project on Github. https://github.com/ML7ru3/2023.2_Project1_MitreAttack.

12 SonarSource. SonarQube. https://www.sonarsource.com/products/sonarqube/

13 Change Visions. Astah. https://astah.net/