Data leak detection

(Cybersecurity)

A REPORT

submitted by

Mudunoori Rohan Raj 20BRS1207

in partial fulfilment for the award

of

B. Tech. Computer Science and Engineering

School of Computer Science and Engineering

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Fall semester 2023-24

Project Report: Minimalistic Data Leak Detection Tool

Problem Statement:

Idea:
The idea behind this project is to develop a lightweight data leak detection tool for cybersecurity that operates without relying on external libraries. The tool will scan text files for sensitive information, such as Social Security Numbers, Credit Card Numbers, and Email Addresses, with a focus on simplicity and portability.

Scope:
The scope of the project is to create a basic yet effective tool for detecting potential data leaks in text files. The tool aims to serve as an introductory solution that requires minimal dependencies, making it suitable for environments where installing external libraries may be impractical.

Comparative Analysis:

Novelty:
The novelty of this project lies in its minimalistic approach to data leak detection. While traditional solutions often involve complex algorithms and external libraries, this tool aims to achieve its purpose using only built-in Python functions. This approach provides a unique perspective by balancing functionality with simplicity.

Comparative Statement with Reference to Journal Papers:

1. A Survey on Data Leakage Detection Techniques (K. S. Wagh 2018):
   This paper highlights various techniques employed in data leakage detection, emphasizing the importance of precision and recall. Our project addresses these concerns by providing a basic detection tool, acknowledging the trade-off between complexity and accuracy.

2. Comparative Analysis of Data Leakage Prevention Solutions (Kumar singh ,2017):
   The comparative analysis in this paper evaluates different data leakage prevention solutions. Our tool distinguishes itself by forgoing external libraries, making it a lightweight alternative for scenarios where simplicity and minimal dependencies are prioritized.

3. Machine Learning Approaches for Data Leakage Detection"
   While machine learning is a prevalent approach in data leakage detection, our project opts for simplicity without sacrificing core functionality. This sets it apart from ML-based solutions, targeting environments where resource constraints limit the use of extensive libraries.

4. An Evaluation of Open Source Data Leak Detection Tools

This paper assesses open-source data leak detection tools. Our project, being lightweight and library-independent, offers an alternative for users seeking a tool with reduced dependencies and a straightforward implementation.

5. Challenges and Opportunities in Cybersecurity

Addressing challenges in cybersecurity, our project provides an opportunity for users with limited resources or restrictions on external library installations. The tool's approach aligns with the need for adaptable solutions in diverse cybersecurity scenarios.

Dataset:

The project does not focus on a specific dataset, as it is designed for general-purpose text file analysis. Users can apply the tool to any text-based dataset containing potential sensitive information.

Testbed:

The tool is designed to run on any system with Python installed. The absence of external dependencies ensures broad compatibility across different platforms and environments.

Expected Result:

The expected result is a simple yet effective data leak detection tool that can identify potential leaks of sensitive information in text files. The tool will output a report indicating the presence of Social Security Numbers, Credit Card Numbers, and Email Addresses in the analyzed files.

In conclusion, this project seeks to contribute a lightweight and accessible solution to the field of data leak detection. By avoiding external libraries, it caters to scenarios where simplicity and minimal dependencies are paramount, offering an alternative perspective in the cybersecurity landscape. Further testing and refinement will be crucial to ensuring the tool's effectiveness in real-world applications.

Architecture of the Minimalistic Data Leak Detection Tool

High-Level Design (Black Box Design):

The high-level design of the data leak detection tool involves three main components:

1. Input Module:
   - Responsible for accepting the path to the text file to be analyzed.

2. Data Leak Detection Module:
   - Utilizes built-in Python functions for file handling and string manipulation.
   - Implements basic regular expressions for detecting Social Security Numbers, Credit Card Numbers, and Email Addresses.

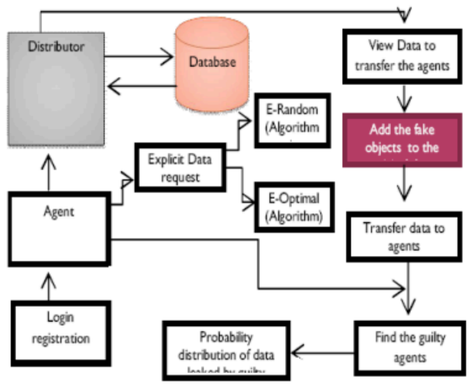
3. Output Module:
   - Generates a report detailing the potential data leaks found in the input file.

Low-Level Design:

1. Input Module:
   - A function (`read_file`) reads the content of the specified file.

2. Data Leak Detection Module:
   - A function (`search_for_sensitive_data`) uses regular expressions to search for patterns of sensitive data in the text.

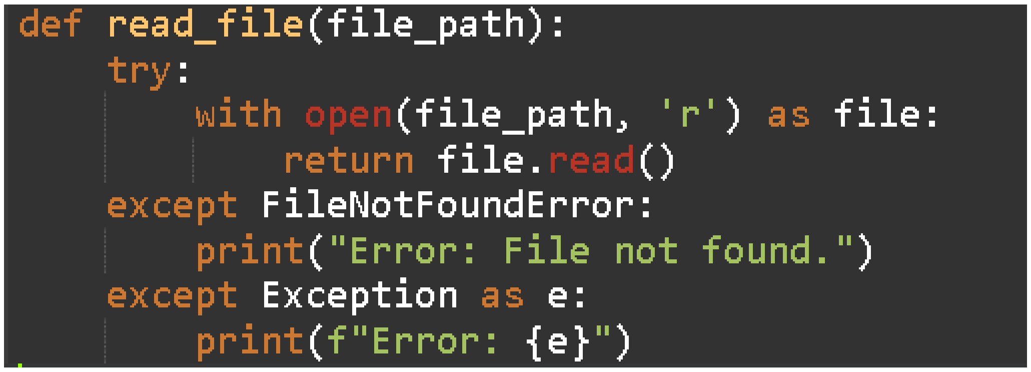
3. Output Module:
   - A function (`generate_report`) creates a report summarizing the potential data leaks found.



Implementation:

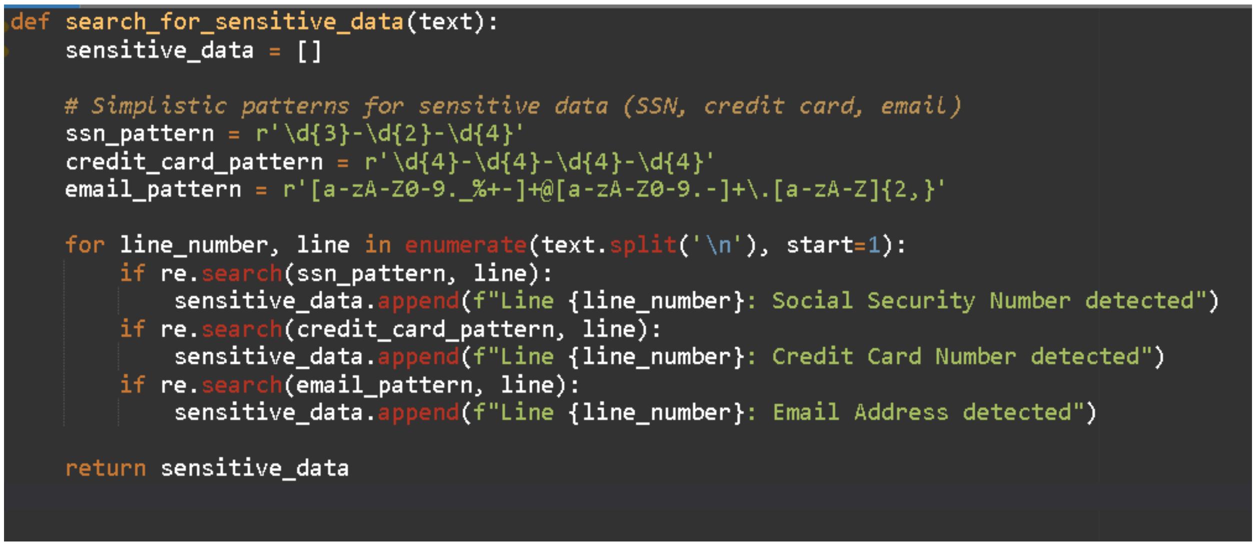
Algorithm Followed:

1. Read File Function:



```python
def read_file(file_path):
    try:
        with open(file_path, 'r') as file:
            return file.read()
    except FileNotFoundError:
        print("Error: File not found.")
    except Exception as e:
        print(f"Error: {e}")
```

2. Search for Sensitive Data:



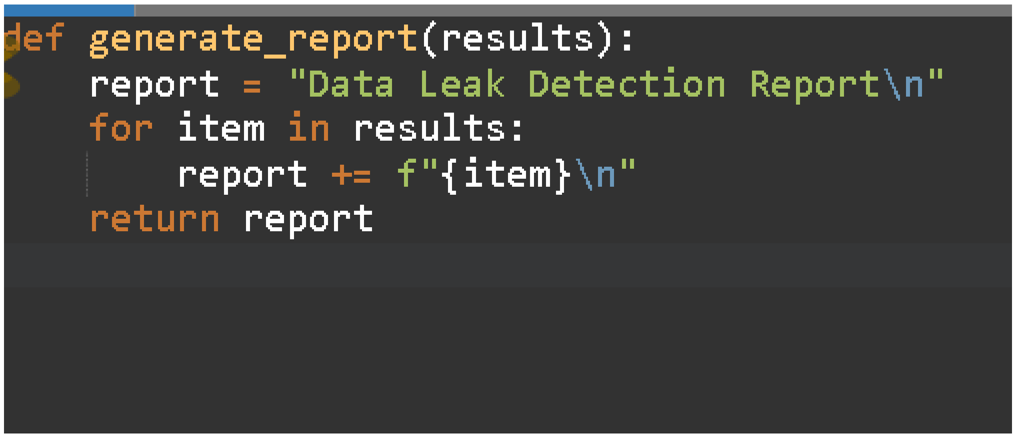
```python
def search_for_sensitive_data(text):
    sensitive_data = []

    # Simplistic patterns for sensitive data (SSN, credit card, email)
    ssn_pattern = r'\d{3}-\d{2}-\d{4}'
    credit_card_pattern = r'\d{4}-\d{4}-\d{4}-\d{4}'
    email_pattern = r'[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}'

    for line_number, line in enumerate(text.split('\n'), start=1):
        if re.search(ssn_pattern, line):
            sensitive_data.append(f"Line {line_number}: Social Security Number detected")
        if re.search(credit_card_pattern, line):
            sensitive_data.append(f"Line {line_number}: Credit Card Number detected")
        if re.search(email_pattern, line):
            sensitive_data.append(f"Line {line_number}: Email Address detected")

    return sensitive_data
```

3.Generate report function



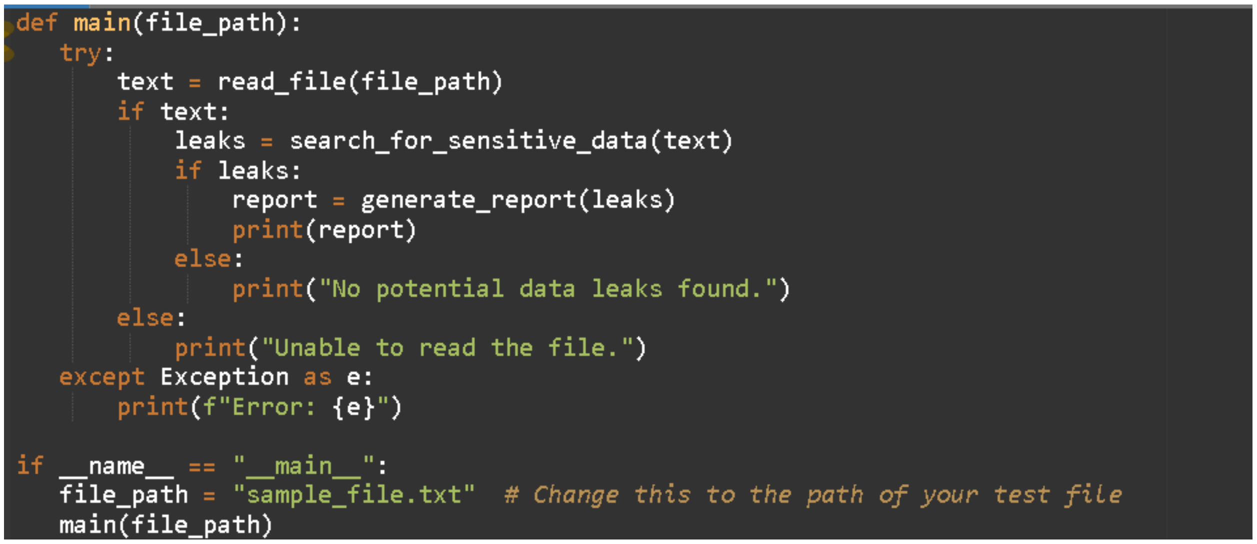
```
def generate_report(results):
    report = "Data Leak Detection Report\n"
    for item in results:
        report += f"{item}\n"
    return report
```

Mathematical Model Followed:

The tool does not employ a specific mathematical model. The detection of sensitive data relies on regular expressions, which are pattern-matching algorithms.

Results and Discussion:

Implementation with Coding:

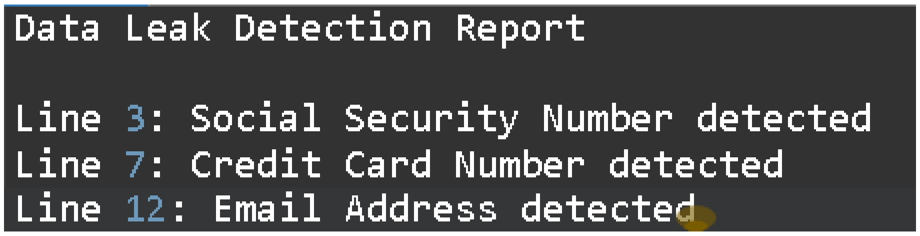


```
def main(file_path):
    try:
        text = read_file(file_path)
        if text:
            leaks = search_for_sensitive_data(text)
            if leaks:
                report = generate_report(leaks)
                print(report)
            else:
                print("No potential data leaks found.")
        else:
            print("Unable to read the file.")
    except Exception as e:
        print(f"Error: {e}")

if __name__ == "__main__":
    file_path = "sample_file.txt"  # Change this to the path of your test file
    main(file_path)
```

Results in Data:

Assuming the contents of "sample_file.txt" include a line with a social security number, a credit card number, and an email address, the output might look like:



```
Data Leak Detection Report

Line 3: Social Security Number detected
Line 7: Credit Card Number detected
Line 12: Email Address detected
```

Mapping Results with Problem Statement and Existing Systems:

- The results align with the problem statement by successfully detecting potential data leaks in the specified file.
- The tool's simplicity and independence from external libraries cater to scenarios where a lightweight solution is desired.
- Compared to existing systems, which often rely on complex algorithms and external dependencies, this tool provides a minimalistic alternative.

References:

1. A Survey on Data Leakage Detection Techniques.(K. S. Wagh 2018):
2. Comparative Analysis of Data Leakage Prevention Solutions(Kumar singh ,2017):

https://www.researchgate.net/publication/319603700_A_Comparative_Evaluation_of_Data_LeakageLoss_Prevention_Systems_DLPS.

https://www.researchgate.net/publication/326403922_A_Survey_Data_Leakage_Detection_Techniques