OS RESOURCE MANAGEMENT AND MALWARE DETECTION SYSTEM

A MINI-PROJECT REPORT

Submitted by

KAVYA VARSHINI H 231901021

LAKSHMI PRIYA P 231901025

in partial fulfillment of the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING (CYBER SECURITY)

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI

BONAFIDE CERTIFICATE

Certified that this project **"OS RESOURCE MANAGEMENT AND MALWARE DETECTION SYSTEM"** is the Bonafide work of **"KAVYA VARSHINI H , LAKSHMI PRIYA P"** who carried out the project work under my supervision.

SIGNATURE

Mrs. JANANEE V

ASSISTANT PROFESSOR

Dept. of Computer Science and Engg,

Rajalakshmi Engineering College

Chennai

This mini project report is submitted for the viva voce examination to be held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

This project is a comprehensive implementation of a Network Security and Operating System (OS) based system that offers both malware detection and system performance monitoring. The purpose of this dual-module system is to strengthen file-level security using YARA-based malware detection, email alerting, file integrity verification, and report generation, while also simulating and analyzing different CPU scheduling algorithms for educational and optimization insights. The project is designed with simplicity, effectiveness, and hands-on learning in mind.

The Network Security (NS) component of the project is responsible for file scanning using YARA rules, monitoring file integrity through SHA-256 hashing, generating PDF reports, and sending real-time alerts when a threat is detected. The Operating System (OS) component implements popular scheduling algorithms like FCFS, SJF, Priority Scheduling, and Round Robin. These are simulated and compared based on turnaround time to suggest the most efficient strategy

ACKNOWLEDGEMENT

We express our sincere thanks to our honorable chairman **MR. S. MEGANATHAN** and the chairperson **DR. M.THANGAM MEGANATHAN** for their timely support and encouragement.

We are greatly indebted to our respected and honorable principal **Dr. S.N. MURUGESAN**

for his able support and guidance.

No words of gratitude will suffice for the unquestioning support extended to us by our Head Of the Department **Mr. BENIDICT JAYAPRAKASH NICHOLAS M.E Ph.D.,** for being ever supporting force during our project work.

We also extend our sincere and hearty thanks to our internal guide **Mrs**.**V JANANEE,** for her valuable guidance and motivation during the completion of this project.

Our sincere thanks to our family members, friends and other staff members of computer science engineering.

1. KAVYA VARSHINI H

2. LAKSHMI PRIYA P

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The increasing complexity and frequency of cyber-attacks and system mismanagement has made it critical to develop and understand the fundamentals of both system security and performance optimization. The goal of this project is to demonstrate key principles from the fields of Network Security and Operating Systems through a beginner-friendly, functional software prototype. This project comprises two core modules:

- Malware Detection and Security Monitoring (NS)
- CPU Scheduling Simulation (OS)

These modules are bundled into a unified runner script, showcasing how real-time malware detection and core OS concepts like scheduling can coexist within one system.

1.2 SCOPE OF THE WORK

- Detect potential malware using custom YARA rules.
- Verify the integrity of files through hash comparison.
- Automatically generate alerts and reports.
- Simulate various CPU scheduling algorithms.
- Compare the efficiency of these algorithms.
- Display results visually in the terminal and generate reports.

1.3 PROBLEM STATEMENT

- How can file-level threats be detected efficiently using pattern-matching techniques?
- How can integrity of important files be monitored?
- Which CPU scheduling algorithm performs the best in various scenarios?
- Can we simulate both domains (NS and OS) in a single combined project?

1.4 AIM AND OBJECTIVES OF THE PROJECT

- To simulate malware detection using YARA rules.
- To perform integrity verification of system files using SHA-256.

- To send email alerts on malware detection.
- To generate detailed PDF reports of malware activity.
- To implement and analyze OS scheduling algorithms.
- To allow users to compare the performance of algorithms visually.

CHAPTER 2

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS

- Processor: Intel Core i5 / Apple M1 or higher
- RAM: Minimum 4 GB
- Hard Disk: 250 GB
- Display: 13” or larger
- Operating System: macOS/Linux preferred (Windows optional)

2.2 SOFTWARE SPECIFICATIONS

- Programming Language: Python 3.9+
- IDE: Visual Studio Code
- Libraries: fpdf, yara-python, smtplib, hashlib, os
- Email Provider: Gmail with App Password
- PDF Generation: FPDF library

CHAPTER 3

MODULE DESCRIPTION

3.1 NETWORK SECURITY MODULE

3.1.1 YARA-Based Malware Detection

YARA rules are created to identify file patterns typically associated with malware. The scanner loads these rules and checks all files in the samples/ directory.

3.1.2 File Integrity Checker

Calculates SHA-256 hashes of files and compares them to stored values to detect tampering or unauthorized changes.

3.1.3 Email Alert System

Sends automated alerts via email when malware is detected.

3.1.4 PDF Report Generator

Creates a summary report of all scanned files, highlighting infected files and their hash values.

3.2 OPERATING SYSTEM MODULE

This module is the core of the project and contains implementations of scheduling algorithms.

3.2.1 FCFS (First Come First Serve):

Processes are scheduled in order they arrive.

3.2.2 SJF (Shortest Job First):

The scheduler chooses the shortest job available next.

3.2.3 Priority Scheduling:

Each process is assigned a priority. The CPU schedules the highest priority process next.

3.2.4 Round Robin:

Assigns fixed time quantum to each process cyclically.

CHAPTER 4

CODING

4.1 FILE STRUCTURE

```
project/
├── rules/
│   └── fake_rules.yar
├── samples/
│   ├── clean_file.txt
│   ├── fake_malware1.txt
│   └── fake_virus.exe
├── src/
│   ├── main.py
│   ├── run_all.py
│   ├── utils/
│   │   ├── integrity_checker.py
│   │   └── report_generator.py
│   ├── email_alert.py
```

│ ├── yara_scanner.py

│ ├── os_algorithms/

│ │ ├── fcfs.py

│ │ ├── sjf.py

│ │ ├── priority_scheduling.py

│ │ └── round_robin.py

4.2 OS SCHEDULING CODE SNIPPETS

4.2.1 FCFS Scheduler:

```
def fcfs(processes):
    processes.sort(key=lambda x: x[1])
    current_time = 0
    result = []
    for pid, arrival, burst in processes:
        start = max(current_time, arrival)
        end = start + burst
        result.append((pid, arrival, start, burst, end))
        current_time = end
    return result
```

4.2.2 SJF Scheduler:

```
def sjf(processes):
    processes.sort(key=lambda x: (x[1], x[2]))
    n = len(processes)
    completed, current_time = 0, 0
    visited = [False]*n
    result = []

    while completed != n:
        idx = -1
        min_burst = float('inf')
        for i in range(n):
            if processes[i][1] <= current_time and not visited[i]:
                if processes[i][2] < min_burst:
                    min_burst = processes[i][2]
```

```
                idx = i
        if idx != -1:
            pid, arrival, burst = processes[idx]
            start = current_time
            end = start + burst
            result.append((pid, arrival, start, burst, end))
            visited[idx] = True
            current_time = end
            completed += 1
        else:
            current_time += 1
    return result
```

4.2.1 Priority Scheduling:

```
def priority_scheduling(processes):
    processes.sort(key=lambda x: (x[1], x[3]))
    current_time = 0
    result = []
    for pid, arrival, burst, priority in processes:
        start = max(current_time, arrival)
        end = start + burst
        result.append((pid, arrival, start, burst, end, priority))
        current_time = end
    return result
```

4.3 NETWORK SECURITY SNIPPETS

YARA Scanner

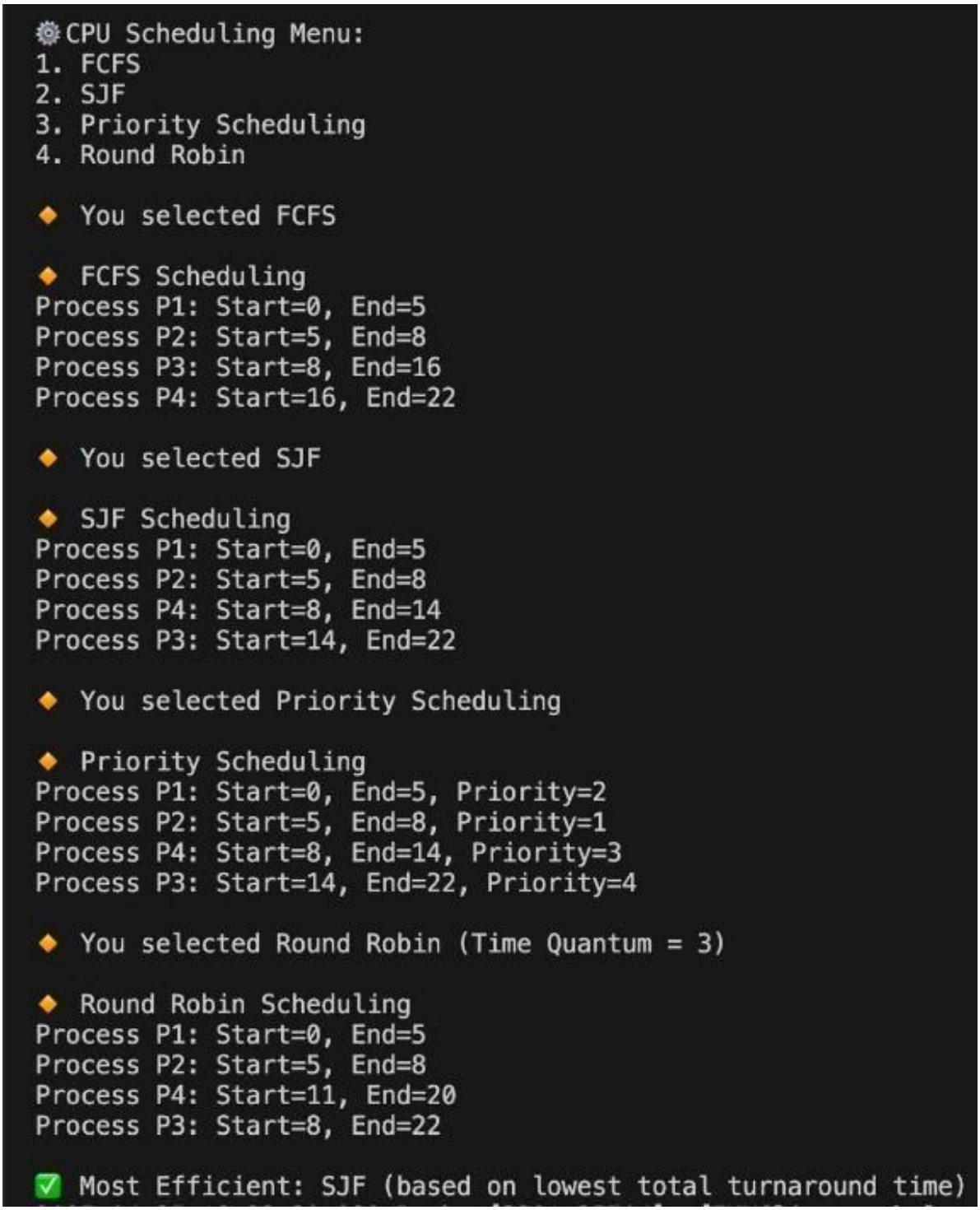
```
import yara

def scan_file(file_path, rule_path="rules/fake_rules.yar"):
    rules = yara.compile(filepath=rule_path)
    matches = rules.match(file_path)
    return matches
```

Email Alert

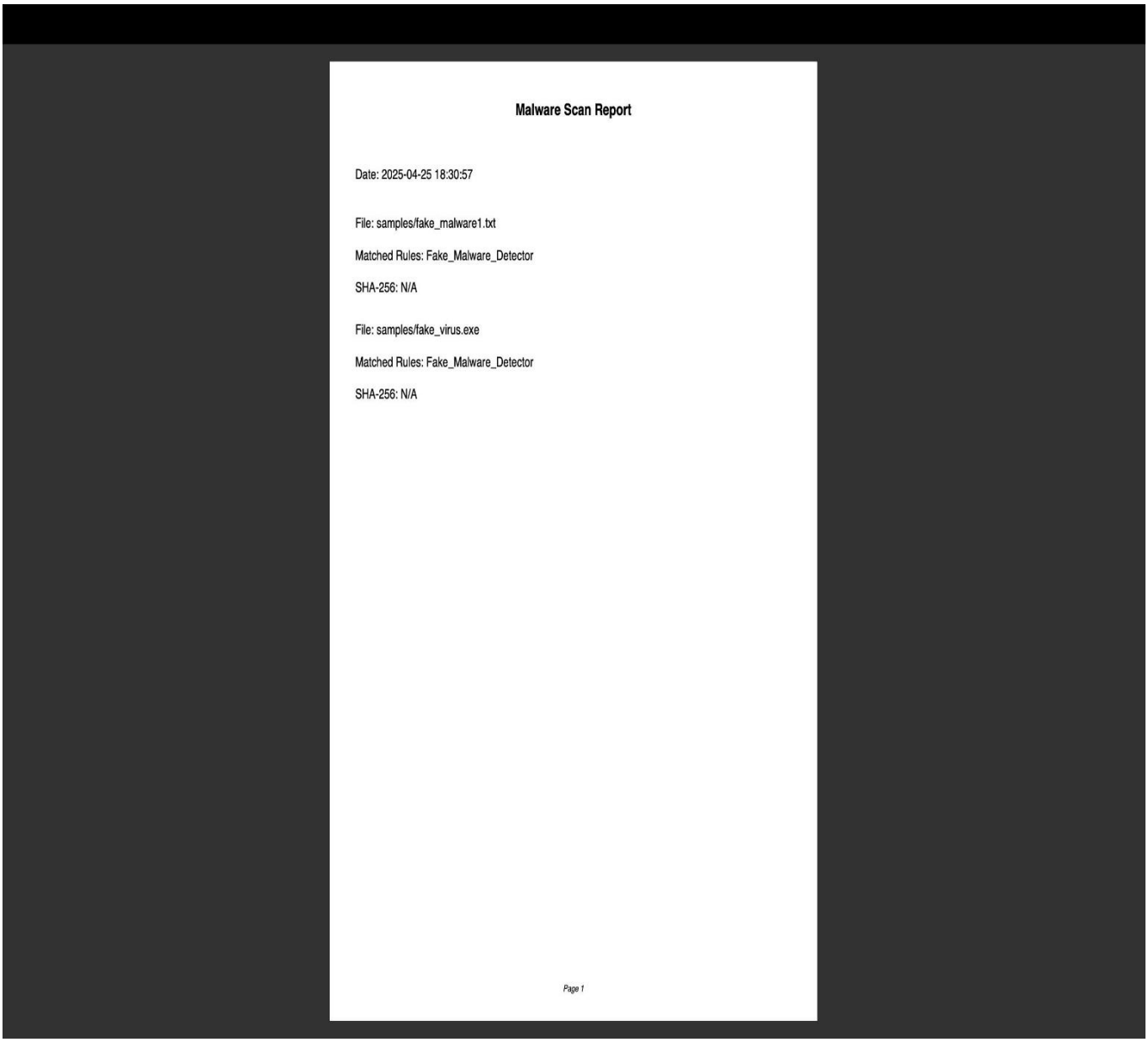
```
def send_email_alert(file_name, match_rule):
    # SMTP configuration and alert logic here
    pass
```

CHAPTER 5

SCREENSHOTS



MALWARE DETECTED TERMINAL OUTPUT



Malware Scan Report

Date: 2025-04-25 18:30:57

File: samples/fake_malware1.txt

Matched Rules: Fake_Malware_Detector

SHA-256: N/A

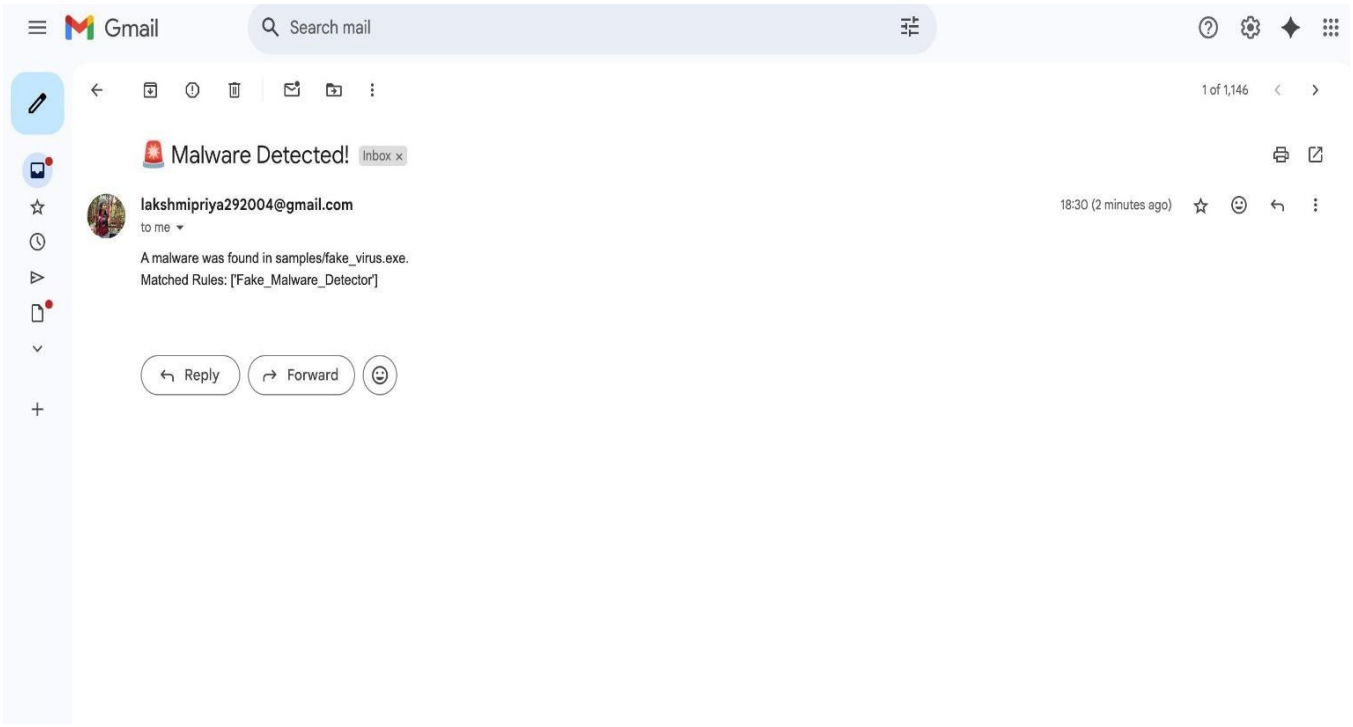
File: samples/fake_virus.exe

Matched Rules: Fake_Malware_Detector

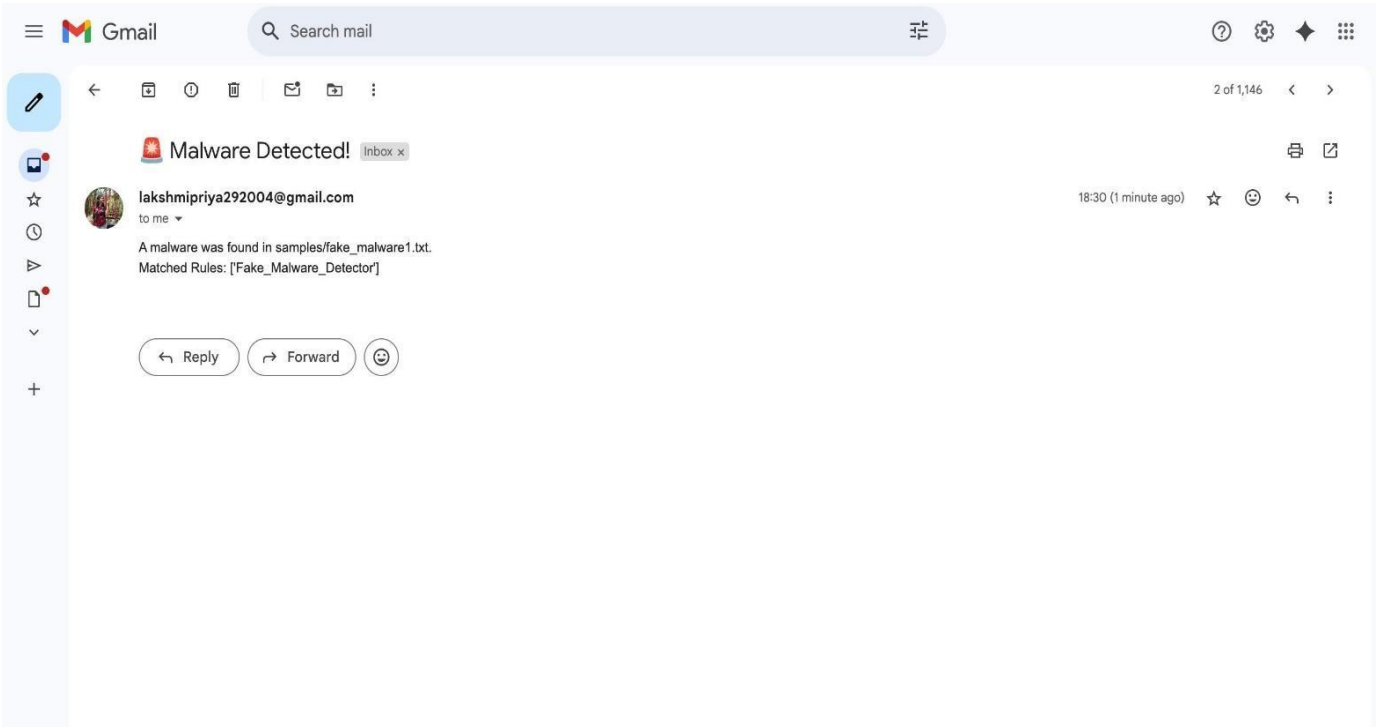
SHA-256: N/A

Page 1

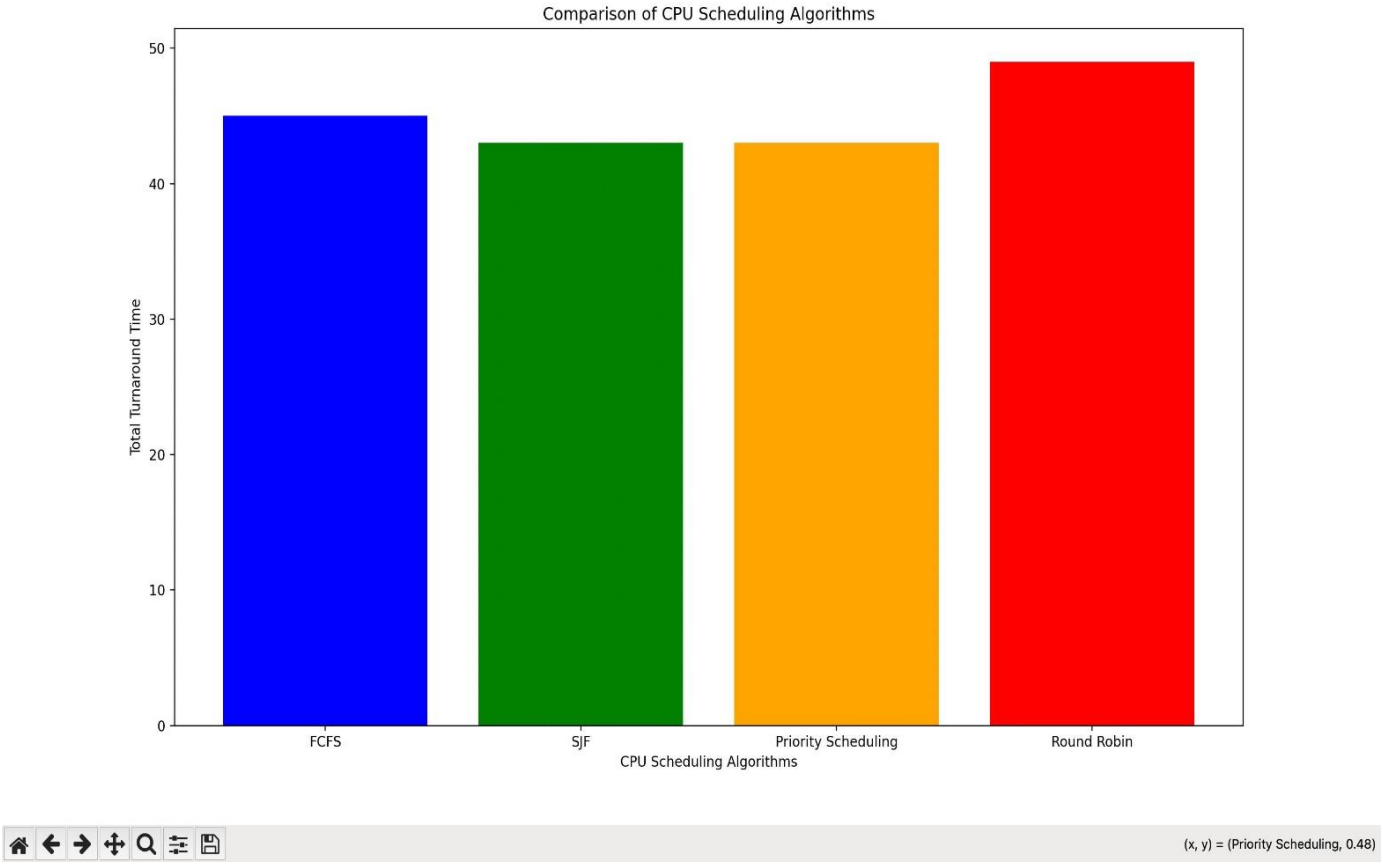
MALWARE SCAN REPORT



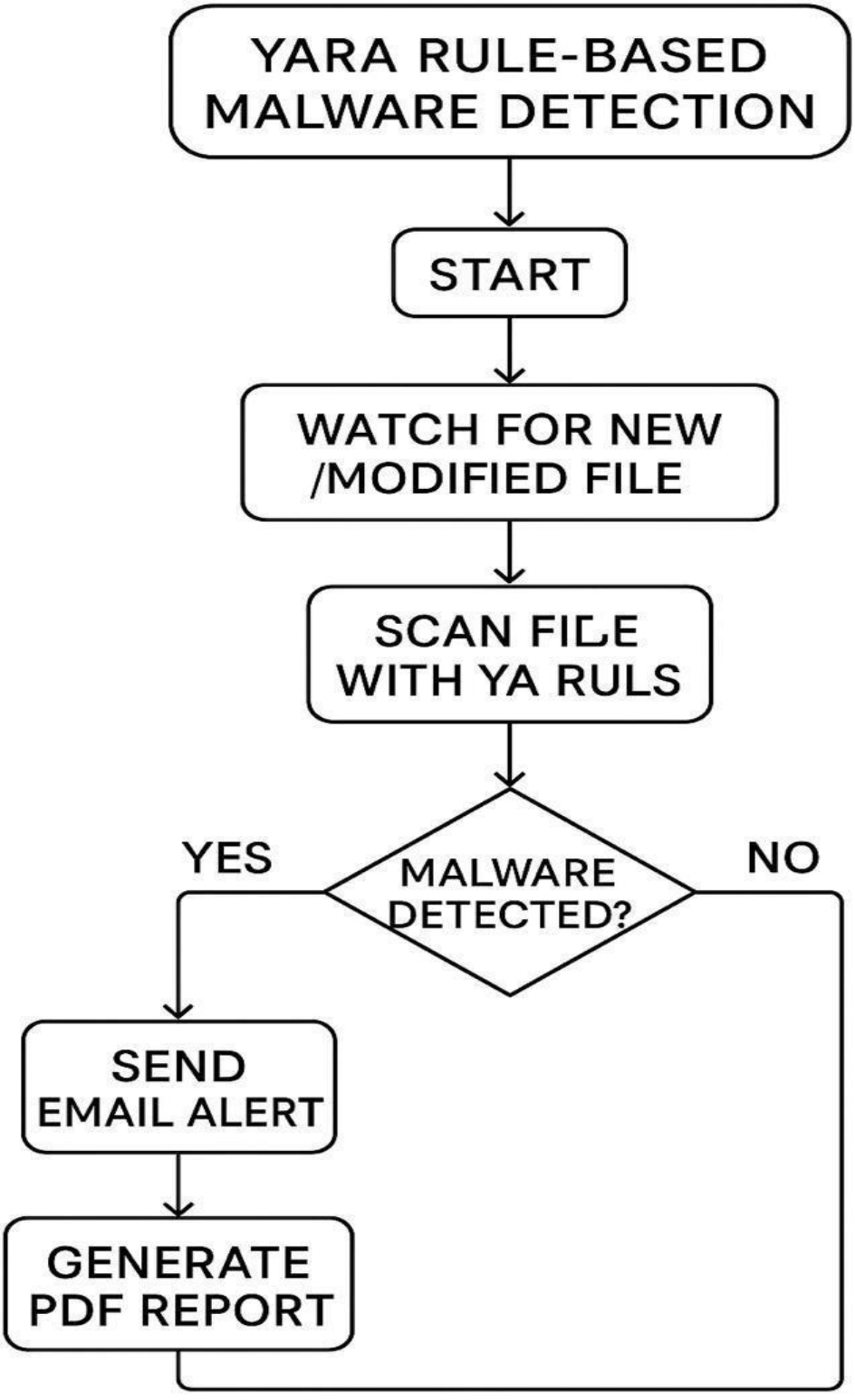
MALWARE DETECTED IN VIRUS.EXE FILE



MALWARE DETECTED IN MALWARE.TXT FILE



COMPARISON OF CPU SCHEDULING ALGORITHMS


YARA RULE-BASED MALWARE DETECTION
START
WATCH FOR NEW /MODIFIED FILE
SCAN FILE WITH YA RULS
MALWARE DETECTED?
YES
NO
SEND EMAIL ALERT
GENERATE PDF REPORT

FLOW CHART

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

This project successfully demonstrates real-time malware detection, file integrity validation, PDF reporting, and CPU scheduling analysis in a unified platform. It integrates core concepts from Network Security and Operating Systems, giving students and beginners hands-on experience with real-world tools and techniques.

FUTURE ENHANCEMENT:

- Extend YARA rules with more advanced malware signatures.
- Use threading to speed up malware scanning.
- Implement a GUI dashboard using Tkinter or Flask.
- Store integrity and scan logs in a SQLite or MongoDB database.
- Add Disk Scheduling and Memory Management simulations.

CHAPTER 7

REFERENCES

1. YARA Documentation- https://yara.readthedocs.io/
2. Python hashlib module- https://docs.python.org/3/library/hashlib.html
3. Python smtplib for sending emails- https://docs.python.org/3/library/smtplib.html
4. fpdf Documentation- https://pyfpdf.github.io/
5. Operating System Concepts by Silberschatz, Galvin, Gagne
6. Network Security Essentials by William Stallings
7. Real Python Tutorials- https://realpython.com/