..

PHISHING DETECTION

A MINI PROJECT REPORT

Submitted by

PREM ROSHAN P	**231901036**
LAKSHMINARAYANAN K	**231901027**
ARAVIND S	**231901006**

in partial fulfillment of the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI

APRIL 2025

BONAFIDE CERTIFICATE

Certified that this project **"PHISHING DETECTION"** is the bonafide work of **"PREM ROSHAN P, LAKSHMI NARAYANAN K, ARAVIND S"** who carried out the project work under my supervision.

SIGNATURE

Mrs. V JANANEE

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

Phishing attacks are a major threat to internet users, aiming to steal sensitive information such as login credentials, bank details, and personal data by impersonating trusted entities. Attackers create fake websites or send fraudulent emails that appear legitimate, making it difficult for users to distinguish between genuine and malicious communications. As the number of phishing incidents continues to grow, there is a pressing need for an effective and user-friendly detection system.

This project proposes a Phishing Detection System that identifies phishing threats based on rule-based analysis and URL inspection techniques. Instead of relying on complex machine learning models, the system utilizes a set of predefined rules and feature checks to detect suspicious patterns in URLs. Some of the features analyzed include URL length, presence of special characters such as '@' or '-', use of IP addresses instead of domain names, multiple redirects, and whether the website uses HTTPS encryption. By evaluating these characteristics, the system can make a quick decision about the likelihood of a URL being malicious.

ACKNOWLEDGEMENT

We express our sincere thanks to our beloved and 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** for being an 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. **PREM ROSHAN P**
2. **LAKSHMI NARAYANAN K**
3. **ARAVIND S**

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
1	**INTRODUCTION**	**7**
1.1	Introduction	**7**
1.2	Scope of the Work	**7**
1.3	Problem Statement	**7**
1.4	Aim and Objectives of the Project	**7**
2	**SYSTEM SPECIFICATIONS**	**8**
2.1	Hardware Specifications	**8**
2.2	Software Specifications	**8**
3	**MODULE DESCRIPTION**	**9**
4	**CODING**	**10**
5	**SCREENSHOTS**	**16**
6	**CONCLUSION AND FUTURE ENHANCEMENT**	**18**
7	**REFERENCES**	**19**

LIST OF FIGURES

Figure No.	Title	Page No.
5.1	Home Page	15
5.2	URL Input and Result Display Page	15
5.3	URL Feature Extraction Example	16
5.4	Rule-based URL Analysis Flowchart	16
5.5	Warning Message for Suspicious URL	17
5.6	Database of Blacklisted URLs	17

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Phishing is a deceptive technique where attackers attempt to gather personal information by imitating trusted sources. Victims are tricked into clicking on malicious links or filling out fake forms. To counter this, the project proposes a real-time Phishing Detection System that uses rule-based logic to assess a given URL's authenticity and check the host system's behavior. This tool is essential for everyday users who might lack the technical know-how to identify phishing threats. It enhances security by offering a proactive way to evaluate suspicious links.

1.2 SCOPE OF THE WORK

The scope of the work encompasses the development of a comprehensive system designed to detect phishing URLs in real-time using non-machine learning (non-ML) rule sets. This system will also include live monitoring of critical system resources such as CPU usage, memory consumption, and disk activity to ensure optimal performance. Additionally, it will generate risk scores based on multiple factors to assess the severity of detected threats. To enhance usability, the system will feature a user-friendly interface, enabling seamless interaction for users of varying technical expertise

1.3 AIM AND OBJECTIVES OF THE PROJECT

Aim:

To develop a phishing URL detection system using rule-based feature analysis for accurate identification of malicious websites.

Objectives:

- Detect phishing URLs using rule-based feature analysis
- Monitor system resources (CPU, memory) during scans.
- Generate user-friendly risk reports.

CHAPTER 2

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS

Component	: Specification
Processor	: Dual core or Higher
RAM	: 8 GB (Minimum)
Storage	:256 GB SSD / 1 TB HDD

2.2 SOFTWARE SPECIFICATIONS

Operating System	: Windows/Linux/MacOS
Dependencies	: psutil, tkinter, matplotlib
Visualization	: GUI
Languages Used	: Python 3.7+
Libraries	: Flask, psutil,pandas, numpy, etc.
Tools	: VS Code, browser (Chrome/Firefox)

CHAPTER 3

MODULE DESCRIPTION

1. URL ANALYSIS MODULE

- Accepts user input (URL) for inspection.
- Extracts URL components such as domain, subdomain, path, and query.
- Suspicious or uncommon Top-Level Domains (TLDs)
- IP address usage instead of domain names.

2. System Monitoring Module

- Uses psutil to gather process data:
    - CPU usage percentage.
    - Memory (RAM) usage percentage.
    - Disk storage usage percentage.
- Detects abnormal system behavior (high CPU/memory usage) during phishing checks.

3. Risk Score Calculation Module

- Assigns weighted scores based on phishing indicators.
- Computes an overall risk score for the submitted URL.
- Flags URLs as safe, suspicious, or highly risky based on the final score.

4. Web Application Module (Flask)

- Provides a user-friendly web interface.
- Allows users to input URLs and view analysis results.
- Displays real-time system metrics through interactive graphs.
- Returns phishing indicators and the risk score in an easily understandable format.

CHAPTER 4

SOURCE CODE:

main.py

```
from flask import Flask, render_template, jsonify, request
from flask_cors import CORS
import psutil
import re
from urllib.parse import urlparse
from datetime import datetime
app = Flask(__name__)
CORS(app)
def get_system_metrics():
    cpu_percent = psutil.cpu_percent(interval=1)
    memory = psutil.virtual_memory()
    disk = psutil.disk_usage('/')
   return {
        'cpu_usage': cpu_percent,
        'memory_usage': memory.percent,
        'disk_usage': disk.percent,
        'timestamp': datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    }
def analyze_phishing_indicators(url, system_metrics):
    try:
        if not url.startswith(('http://', 'https://'))
        url = 'http://' + url
```

```
        parsed_url = urlparse(url)
        domain = parsed_url.netloc.lower()
        path = parsed_url.path.lower()
        query = parsed_url.query.lower()
    except Exception:
        domain = url.lower()
        path = ''
        query = ''
    legitimate_tlds = {'.com', '.org', '.net', '.edu', '.gov', '.mil', '.int', '.eu', '.us', '.uk', '.ca', '.au', '.de', '.fr', '.jp'
 suspicious_tlds = {'.xyz', '.tk', '.ml', '.ga', '.cf', '.gq', '.pw', '.cc', '.top', '.work', '.party', '.date', '.stream', '.racing', '.win', '.bid', '.loan'}
 tld = '.'+domain.split('.')[-1] if '.' in domain else ''
    brand_names = {
        'google': 'google.com',
        'facebook': 'facebook.com',
        'microsoft': 'microsoft.com',
        'apple': 'apple.com',
        'amazon': 'amazon.com',
        'paypal': 'paypal.com',
        'netflix': 'netflix.com',
        'instagram': 'instagram.com',
        'twitter': 'twitter.com',
        'linkedin': 'linkedin.com'
    }
 indicators = {
        'suspicious_tld': (tld in suspicious_tlds) or (tld not in legitimate_tlds),
       'ip_address': bool(re.match(r'^(?:[0-9]{1,3}\.){3}[0-9]{1,3}(?::[0-9]+)?$', domain)),
```

```
        'suspicious_ports': ':' in domain and domain.split(':')[-1].isdigit() and int(domain.split(':')[-1]) not in [80, 443],
        'system_anomaly': system_metrics['cpu_usage'] > 90 or system_metrics['memory_usage'] > 90,
        'long_url': len(url) > 100,
        'suspicious_chars': bool(re.search(r'[0-9]|[-]', domain.split('.')[0])) if '.' in domain else False,
        'multiple_subdomains': domain.count('.') > 2,
        'suspicious_keywords': any(keyword in (path + query) for keyword in [
            'login', 'signin', 'verify', 'account', 'secure', 'update', 'password',
            'confirm', 'verification', 'authenticate', 'wallet', 'security'
        ]),
        'brand_impersonation': any(
            brand in domain and brand_names[brand] not in domain
            for brand in brand_names
        ),
        'special_chars': bool(re.search(r'[^a-zA-Z0-9.-]', domain)),
        'numeric_domain': bool(re.search(r'^[0-9]+', domain.split('.')[0])),
        'suspicious_patterns': bool(re.search(r'(secure|login|account|update|verify)[0-9]', domain))
    }
    weights = {
        'suspicious_tld': 0.25,
        'ip_address': 0.15,
        'suspicious_ports': 0.10,
        'system_anomaly': 0.05,
        'long_url': 0.05,
        'suspicious_chars': 0.10,
        'multiple_subdomains': 0.05,
```

```
        'suspicious_keywords': 0.10,
        'brand_impersonation': 0.20,
        'special_chars': 0.10,
        'numeric_domain': 0.10,
        'suspicious_patterns': 0.15
    }
    risk_score = sum(indicators[key] * weights[key] for key in weights) * 100
    risk_score = min(max(risk_score, 0), 100)
    high_risk_count = sum(1 for key in ['suspicious_tld', 'ip_address', 'brand_impersonation', 'suspicious_patterns'] if indicators[key])
    if high_risk_count >= 2:
        risk_score = min(risk_score * 1.5, 100)  # Increase score by 50% if multiple high-risk indicators
    return indicators, risk_score
@app.route('/')
def index():
    return render_template('index.html')
@app.route('/api/system-metrics')
def system_metrics():
    return jsonify(get_system_metrics())
@app.route('/api/analyze', methods=['POST'])
def analyze_url():
    data = request.json
    url = data.get('url', '')
    system_metrics = get_system_metrics()
    try:
        indicators, risk_score = analyze_phishing_indicators(url, system_metrics)
```

```
    return jsonify({
            'indicators': indicators,
            'risk_score': risk_score,
            'system_metrics': system_metrics,
            'analyzed_url': url
        })
    except Exception as e:
        return jsonify({
            'error': str(e),
            'risk_score': 0,
            'system_metrics': system_metrics
        }), 400
if __name__ == '__main__':
    app.run(debug=True)
```

run.py

```
from app import app
if __name__ == '__main__':
    print("Starting Flask application...")
    app.run(debug=True, port=5000)
```

process_monitor

```
            import psutil
            def get_processes():
                processes = []
                for proc in psutil.process_iter(['pid', 'name', 'memory_info']):
                    try:
                        pid    =    proc.info['pid']
                        name = proc.info['name']
                        mem_info = proc.info['memory_info']
```

```
            ram_mb = mem_info.rss / 1024 / 1024
            ram_pct = proc.memory_percent()
            cpu_pct = proc.cpu_percent(interval=None)
            processes.append((
                pid,
                name,
                round(ram_mb, 2),
                round(ram_pct, 3),
                round(cpu_pct, 2)
            ))
        except (psutil.NoSuchProcess, psutil.AccessDenied, psutil.ZombieProcess):
            continue
    return processes
```

simple_app.py

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello():
    return 'Hello, World!'
if __name__ ='__main__':
    print("Starting simple Flask application...")
app.run(debug=Trueport=5000)
```

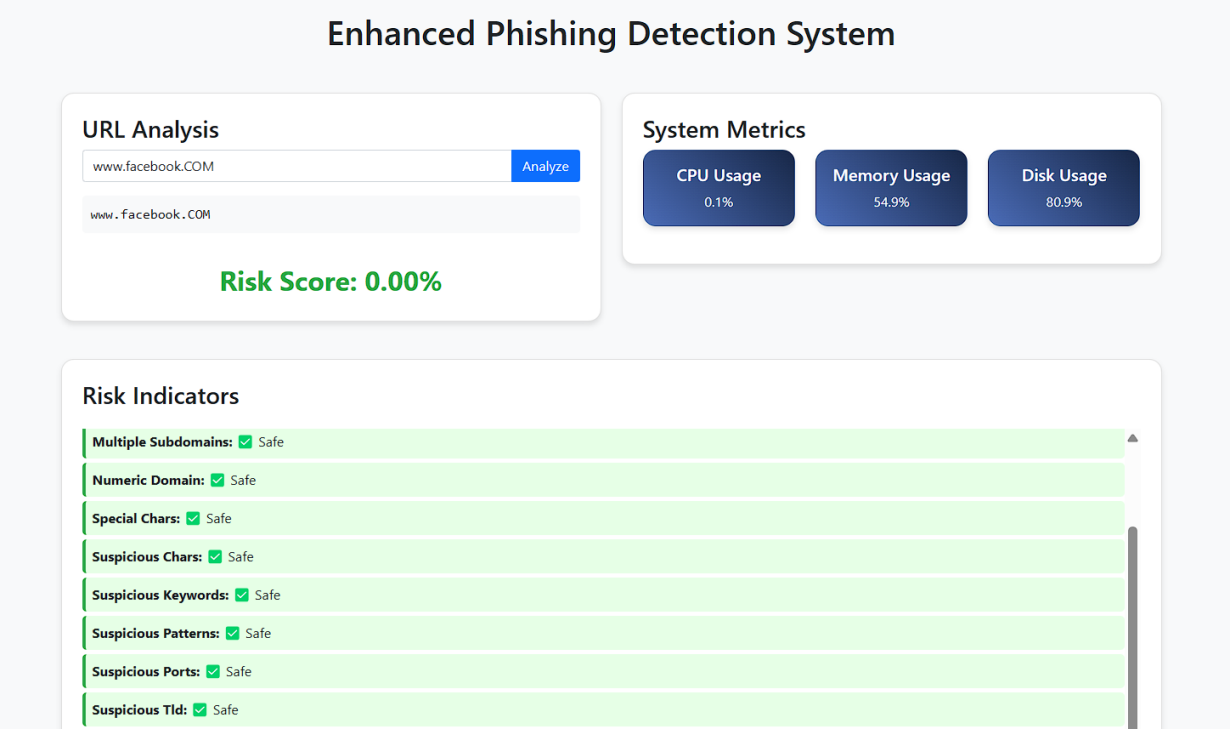
requirements .txt

```
flask==2.0.1
psutil==5.8.0
pandas==1.3.3
numpy==1.21.2
scikit-learn==0.24.2
requests==2.26.0
python-dotenv==0.19.0
flask-cors==3.0.10
```

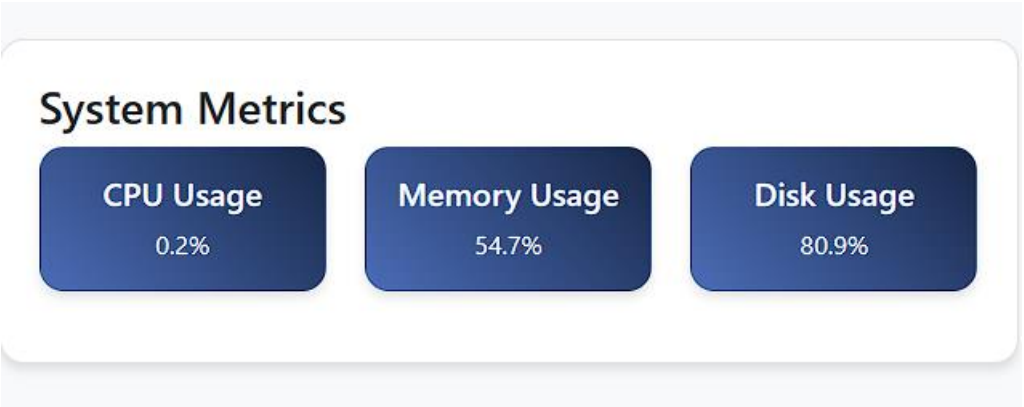
CHAPTER 5

SCREENSHOTS

4. PHISHING DETECTION



4.1 -CPU&DISK RAM USAGE



4.2 CPU & DISK USAGE GRAPH

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

Conclusion:

This project provides an efficient rule-based system to detect phishing threats in real time. Unlike machine learning systems, this approach is lightweight, fast, and transparent in how it calculates risk. The integration of system monitoring adds an additional layer of awareness, allowing users to notice suspicious spikes in system behavior..

Future Enhancements:

- Add real-time email phishing detection.
- Build a browser extension.
- Integrate blacklisted domain database.
- Add SMS (smishing) detection.

CHAPTER 7

REFERENCES

1. • **OWASP Foundation** – Phishing Detection Guidelines
   https://owasp.org/www-community/phishing
2. • **Python Official Documentation** – Regex and URL Parsing
   https://docs.python.org/3/library/re.html
   https://docs.python.org/3/library/urllib.parse.html
3. • **psutil Documentation** – System Monitoring in Python
   https://psutil.readthedocs.io/en/latest/
4. • **Flask Documentation** – Lightweight Python Web Framework
   https://flask.palletsprojects.com/
5. • **Mozilla Developer Network (MDN)** – URL Structure and Security
   https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_URL

   .