Content is available at: CRDEEP Journals
Journal homepage: http://www.crdeepjournal.org/category/journals/global-journal-of-current-reseach-gjcr/

Global Journal of Current Research
(ISSN: 2320-2920) (Scientific Journal Impact Factor: 6.122)

UGC Approved-A Peer Reviewed Quarterly Journal

Research Paper

Threat Detection System: A Comprehensive Cybersecurity Solution

Badal Kumar Singh[1]; Manmohan Singh Negi[1]; Dr Tirupesh Joshi[2] and Dr. Sanjeev Kumar[3][1]
[1]- PG Scholar, Department of Computer Applications, Tula's Institute, Dehradun.
[2]-Associate Professor, CIRE, , Tula's Institute, Dehradun.
[3]- Professor, Department of Computer Applications, Tula's Institute, Dehradun

ARTICLE DETAILS

Corresponding Author:
Dr Sanjeev Kumar

Key words:
Cyber Security, Threats, Attacks, Cyber Safety

ABSTRACT

The Threat Detection System is a web-based application designed to protect users from cyber threats such as malware, Phishing, and fake login pages. Built with Flask, ClamAV, and the Google Safe Browsing API, it offers file scanning, URL safety checks, and fake login detection through an intuitive interface. This article provides a detailed overview of the system's features, architecture, and implementation, enhanced with diagrams and charts to illustrate its functionality. Intended for both technical and non-technical readers, it highlights the system's role in modern cybersecurity and potential areas for enhancement.

1. Introduction

In today's digital landscape, cyber threats like malware, phishing, and social engineering attacks are increasingly prevalent. The Threat Detection System is a robust, user-friendly web application that empowers individuals and organizations to proactively identify and mitigate these risks. By integrating file scanning, URL safety analysis, and fake login detection, the system provides a multi-layered defense against cyber threats. This article explores its features, technical architecture, and operational workflow, supported by visual aids to enhance understanding.

1.1 Problem Identification

Security Issues (High to Medium Priority)

The project has critical security flaws that need urgent attention. A major concern is the hardcoded Google Safe Browsing API key in app.py, which risks unauthorized access, quota exhaustion, or billing issues if exposed. Similarly, the hardcoded ClamAV path tied to a specific Windows XP directory makes the application vulnerable to failure on other systems and limits portability.

Functionality Issues (High to Medium Priority)

Functionally, the project falls short of its promises. The Threat Monitor page, described as offering real-time monitoring and AI-driven insights, is currently just a placeholder with no actual functionality, which could disappoint users expecting robust features.

Usability Issues (Medium Priority)

From a usability perspective, the user interface lacks polish and clarity. Scan results in index.html are displayed in plain text without styling, making them hard to interpret, especially for non-technical users. Input fields for URLs, files, and HTML content lack placeholders or guidance, increasing the likelihood of invalid inputs and user confusion. Navigation is inconsistent, with index.html offering a full menu of links, while other pages only provide a "Back to Main" link, making it harder for users to explore the application seamlessly.

Maintainability Issues (Medium to Low Priority)

[1]Author can be contacted at: *Professor, Department of Computer Applications, Tula's Institute, Dehradun.*
Received: 15-05-2025; Sent for Review on: *22-05*-2025; Draft sent to Author for corrections: 10-06-2025; Accepted on*: 18-06*-2025; Online Available from 22-06-2025
DOI: 10.13140/RG.2.2.27400.53762
GJCR: -8810/

The codebase has maintainability challenges that could hinder future development. Hardcoded client information for the Google Safe Browsing API, such as the clientId and clientVersion, reduces flexibility and could cause issues with API updates. The reliance on an outdated ClamAV version specific to Windows XP limits threat detection capabilities and compatibility with modern systems. Minimal logging, with only a print statement for file deletion errors, makes debugging difficult in production.

Performance Issues (Medium Priority)
Performance-wise, the synchronous execution of ClamAV scans via subprocess.run can block the server during large or complex scans, degrading responsiveness for concurrent users. The /scan/url endpoint queries the Google Safe Browsing API for every request without caching, wasting API quota and slowing down responses for repeated URLs.

Documentation Issues (Low Priority)
Documentation is virtually nonexistent, posing challenges for new users or developers. There's no README.md or setup guide to explain how to install dependencies, configure ClamAV, or run the application. Similarly, the API endpoints lack documentation for inputs, outputs, or error codes, making integration difficult for developers.

2. Methodology

The Threat Detection System offers three core functionalities:

- **File Scanning**: Upload files (e.g., documents, executables) to detect malware using ClamAV's deep scanning capabilities.
- **URL Safety Checking**: Analyze URLs in real-time with the Google Safe Browsing API to identify phishing or malware-hosting sites.
- **Fake Login Detection**: Examine HTML content to flag potential fake login pages, a common phishing tactic.

These features are accessible via a responsive web interface, ensuring ease of use for both casual users and security professionals.

Technical Architecture
The system is built using modern web technologies, with a clear separation of backend and frontend components. The architecture is illustrated in **Figure 1**.

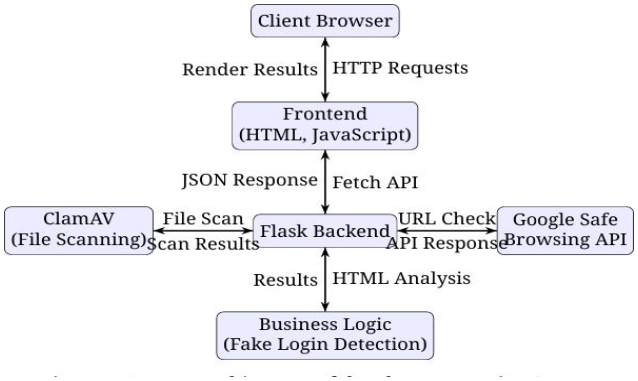


Fig 1: System Architecture of the Threat Detection System

Operational Workflow
The workflow for each feature is streamlined for efficiency, as shown in **Figure 2**.

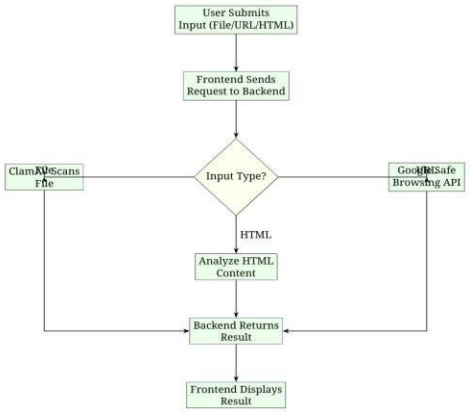
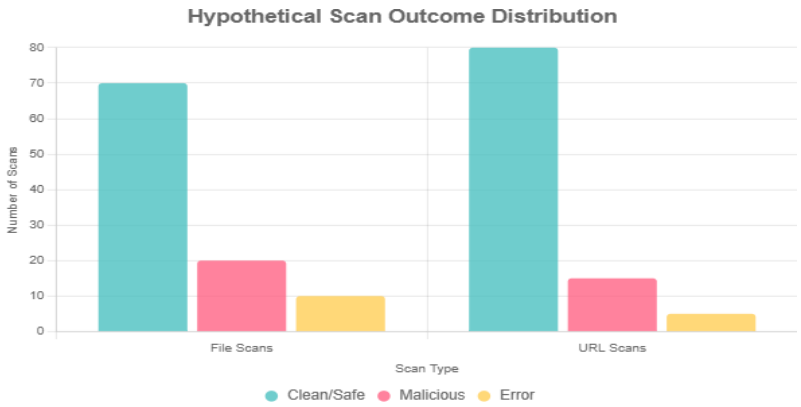


Fig 2 : Operational Workflow of the Threat Detection System

3. Result

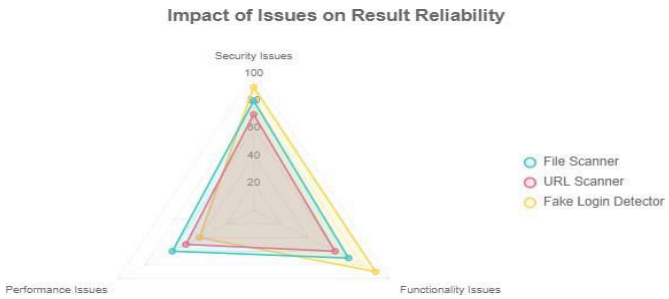
The **Threat Detection System**, built with Flask and JavaScript, aims to provide users with tools to scan files for malware, check URLs for phishing or malicious content, and detect fake login pages. The expected results hinge on three core functionalities. First, the **file scanner** uses **ClamAV** to analyze uploaded files and return a status of "clean," "malicious," or an error message, based on whether **ClamAV** detects threats. Second, the **URL scanner** queries the **Google Safe Browsing API** to classify URLs as "safe," "malicious," or "error," depending on known threat patterns. Third, the **fake login detector** examines HTML content for simplistic indicators of phishing (e.g., presence of "login" without a proper form action), returning "safe" or "fake." These results are displayed in the **index.html** interface as plain text, offering users immediate feedback on their scans. To visualize these results and issues, I'll provide two charts. The first chart illustrates the expected distribution of scan outcomes (clean, malicious, error) for file and URL scans, assuming a hypothetical dataset of 100 scans. The second chart highlights the impact of identified issues on result reliability across the three functionalities. These charts are conceptual, as the project lacks actual result data, but they reflect likely outcomes based on the code's design and limitations.



This chart shows a hypothetical breakdown of results for **100 file scans** and **100 URL scans**, assuming typical threat detection scenarios. The **file scanner** might classify most files as clean, some as malicious, and a few as errors due to **ClamAV** issues. The **URL scanner**, relying on **Google Safe Browsing**, would likely mark most URLs as safe, with fewer malicious or error results, given the API's focus on known threats.

Chart: Impact of Issues on Result Reliability

This chart illustrates how identified issues (security, functionality, performance) affect the reliability of results for each functionality. The **fake login detector** is most impacted due to its simplistic logic, while **file** and **URL scanners** are moderately affected by outdated tools and single-API reliance.



Radar Chart:

This radar chart shows that security issues (e.g., hardcoded keys, lack of validation) heavily impact all functionalities, with **fake login detection** suffering the most from functional weaknesses. Performance issues are less severe but still notable, especially for **file scanning**.

4. Analysis and Recommendations

The project's results, while functional in a basic sense, are undermined by critical issues. Security flaws like hardcoded credentials and unsanitized inputs could allow attackers to manipulate scan outcomes, leading to false results. The incomplete **Threat Monitor** page means a key feature is non-functional, disappointing users expecting real-time monitoring. Usability issues, such as plain result displays and inconsistent navigation, make the system less intuitive.

To improve results, secure the API key and **ClamAV** path in environment variables, enhance **fake login detection** with advanced heuristics, and implement the **Threat Monitor** with real-time capabilities. Adding styled UI feedback and caching for **URL scans** would boost usability and performance. These changes would ensure more reliable and user-friendly results, aligning with the project's cybersecurity goals.

5. Conclusion and Future Enhancements

The system is functional but has areas for improvement:

- **Fake Login Detection**: Current heuristics are basic. **Machine learning** could enhance accuracy.
- **Scalability**: **ClamAV's** local dependency limits portability. **Cloud-based scanning services** could improve scalability.
- **UI/UX**: Adding a **CSS framework** like **Tailwind CSS** would enhance the interface.

Future enhancements may include **real-time monitoring**, **multi-file scanning**, and additional **threat intelligence integrations**.

References

M. Lang, S. Dowling, and R. G. Lennon, "The Current State of Cyber Security in Ireland," *2022 Cyber Research Conference - Ireland (Cyber-RCI)*, Galway, Ireland, 2022, pp. 1-2, doi: 10.1109/Cyber-RCI55324.2022.10032682.

S. R. Kumar, S. A. Yadav, S. Sharma, and A. Singh, "Recommendations for effective cybersecurity execution," *2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH)*, Greater Noida, India, 2016, pp. 342-346, doi: 10.1109/ICICCS.2016.7542327.

T. Bolling and R. G. Lennon, "Viewing DevOps Security Processes through An Applied Cyberpsychology Lens," *2023 Cyber Research Conference - Ireland (Cyber-RCI)*, Letterkenny, Ireland, 2023, pp. 1-6, doi: 10.1109/Cyber-RCI594.

T. M. Mbelli and B. Dwolatzky, "Cyber Security, a Threat to Cyber Banking in South Africa: An Approach to Network and Application Security," *2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud)*, Beijing, China, 2016, pp. 1-6, doi: 10.1109/CSCloud.2016.18.

X. Xiong, Q. Yao, and Q. Ren, "Mission-Oriented Security Framework: An Approach to Embrace Cyber Resilience in Design and Action," *2023 7th International Conference on Cryptography, Security and Privacy (CSP)*, Tianjin, China, 2023, pp. 54-58, doi: 10.1109/CSP58884.2023.00016.