**MALWARE ANALYSIS AND DETECTION SYSTEM**

KIBABII UNIVERSITY

ALEX MWANGI

COM/0027/22

30th JAN 2025

**Introduction**

The digital world is facing several challenges regarding cyber threats. With the rapid increase in malware attacks, these attacks have become more complex. Organizations and individuals face a number of risks: data breach, financial loss of systems caused by malicious software. In this regard, an efficient malware analysis and detection system is required that can help find and analyze suspicious files as well as store information about them.

This project proposes, designs, and development of a malware analysis and detection system using Python-Django and JavaScript. It allows the web user to upload files; based on that, the uploaded files will be analyzed against known threats. Initial work would start with static file analysis; thus, when a file gets uploaded, it gets scanned for known signatures, metadata, and patterns that may be suspicious. The results of this shall be maintained in a structured database for future reference and threat intelligence.

The project will expose an intuitive, web-based interface that allows users to upload files for analysis and view reports on those analyses. The back-end, powered by Django, will be responsible for processing the files, managing databases, and storing results. JavaScript will be used for an interactive front-end experience. Further into the project, additional features such as behavioral analysis, sand-boxing and API integrations may be considered.

### ****Problem Statement****

The increasing number of **malware attacks** poses a significant threat to individuals, businesses, and organizations. Cybercriminals use malicious files to steal sensitive data, disrupt systems, and gain unauthorized access to networks. Traditional antivirus solutions may not always detect emerging threats, making it essential to have **custom malware analysis tools** that provide deeper insights into suspicious files.

Currently, many available malware detection systems are either **too complex, resource-intensive, or expensive**, making them inaccessible to students, researchers, and small organizations. Additionally, most traditional scanners focus on **signature-based detection**, which may not identify new or modified malware strains.

This project aims to address these challenges by developing a **lightweight, web-based Malware Analysis and Detection System** that allows users to upload and analyze files for potential threats. The system will focus on **static analysis**, checking file metadata, hashes, and signatures against known malicious patterns. The results will be stored in a structured database for further reference.

By providing an **easy-to-use, cost-effective, and scalable** malware detection tool, this project will help users **identify suspicious files**, store results for future investigation, and lay the foundation for more advanced threat analysis in the future.

### ****Specific Objectives****

****Develop a Web-Based Malware Analysis System:** The web-based malware analysis system should be developed using Django-a Python framework-and JavaScript to provide an interactive web application that allows users to upload files for analysis.**

****File Upload and Secure Storage:** Allow the web user to submit the files through the web interface and securely store them in a structured database.**

****Static File Analysis:** Extract the metadata of the files classified and compute hash values-MD5, SHA-256-and compare it against the malware signatures.**

****Classify the files based on the threat level** by categorizing them into safe, suspicious, and malicious using predefined rules and detection patterns.**

****Store and Manage Analysis Reports:** Store the results of the scan into a database for future reference, retrieval, and organization.**

****Provide a Simple and User-Friendly Interface:** Design an intuitive front-end for users to easily upload files, view reports, and interact with the system.**

****Security and Efficiency of the System:** Basic validation of files, access controls, and efficient handling of databases will provide integrity to the system.**

**Purpose of the project**

The purpose of the **Malware Analysis and Detection System** project is to provide an accessible, lightweight tool for detecting and analyzing potentially malicious files. The system aims to offer basic static analysis by examining file metadata, hash values (e.g., MD5, SHA-256), and comparing these against known malware signatures to classify files as safe, suspicious, or malicious. This is especially beneficial for students, researchers, and cybersecurity enthusiasts who need a cost-effective solution for basic malware detection.

The project aims to:

**Enable file analysis**: Allow users to upload files for malware inspection, providing insights into the file's nature.

**Support Cybersecurity research**: Offer a foundational tool for researchers to understand malware patterns and trends.

**Improve access to malware detection**: Provide an affordable and user-friendly alternative to complex, commercial malware analysis tools.

**Create a scalable foundation for future development**: Lay the groundwork for incorporating advanced techniques like sandboxing, behavioral analysis, and machine learning-based detection in future iterations.

### Project Scope

The project designed and implemented is a Malware Analysis and Detection System focused on the scope of static file analysis and secure storage. Users can upload files in the system; it will check for any type of threat and store data about such threats for future reference.

**In-Scope (What the Project Will Cover)**

**1. File Upload and Storage:** Users will be able to upload files through a web-based interface. The system will securely store both the uploaded files and their analysis results.

**2. Static Analysis:** The system will analyze files using techniques such as: Extracting metadata: file type, size and structure.

**3. Computing file hashes:** MD5, SHA-256, by comparing against known malware signatures. Scanning for suspicious patterns or embedded scripts.

**4. Database Management:** All uploaded files and their analysis reports would be stored in a structured database, which could facilitate easy access and reference.

**5. User Interface:** A straightforward, intuitive web interface using Django for uploading files and viewing reports.

**6. Basic Threat Classification:** Based on the detection of risks, the files would be categorized into 'safe', 'suspicious', or 'malicious'.

**Out of Scope (What the Project Will Not Cover)**

**1. Advanced AI-Based Detection:** The project will not include machine learning models for threat detection at this stage.

**2. Behavioral Analysis (Sandboxing):** The system will not execute files in a controlled environment to monitor their behavior.

**3. Real-Time Malware Detection:** The project will focus on on-demand file scanning rather than live system protection.

**4. Integration of Threat Intelligence Feeds:** There will be no integration with external malware databases and APIs.

### **Methodology**

**1. Requirement Analysis**

**Identify Stakeholders:** Target audience of the system includes students, researchers, and small organizations showing huge interest in cybersecurity.

**Gather Requirements:** Requirements include being able to upload files for analysis, receive reports back, light weightiness, security in the system, ease of use among others.

**Functional description:** It needs to have uploading, malware detecting, categorizing files, generating reports.

**2. System design**

**Architecture Design:** The design follows a client-server architecture using web-based HTML/CSS/JS on the client side, supported by the Python Django back end for the entire process of handling file uploads to processing and report generation.

**Database Design:** Design the relational database, by using SQLite, that will be used for file metadata, analysis results, user information, and logs.

**UI/UX Design:** Wireframe and prototype the user interface to ensure it's going to be intuitive and user-friendly.

**3. Development**

**Frontend Development:** Design a user interface that is interactive using HTML5, CSS3, and JavaScript (React or vanilla JS) for file upload, report view, and interacting with the system.

**Backend Development:** It will be done with Django for back-end purposes, handling file uploads, file analysis, and database interactions. Third-party Python libraries for file static analysis include hashlib and pefile.

**File Analysis Implementation:** Implement static file analysis techniques, including hash value generation (MD5, SHA-256) and pattern matching using YARA rules to detect known malware signatures.

**Database Implementation:** Create database models for storing analysis results, user data, and metadata associated with the uploaded files.

**4. Testing**

**Unit Testing:** Test the individual components, such as file uploading, computation of the hash, matching of YARA rules, and all the other database operations to ensure that they will behave as expected.

**Integration Testing:** Perform testing of the entire system to ensure that the frontend, backend, and the database are all functioning as one.

**Security Testing:** Perform security testing, which includes file validation for preventing malicious files from corrupting the system and ensuring secure file uploads.

**5. Deployment**

**Deployment Setup:** Deploy the system to a cloud platform, say AWS, in order to make it accessible to users. Use Amazon S3 for storing uploaded files securely.

**Server Configuration:** The installation of the server will include all performance enhancements and configuration of SSL certificates to establish secure connections over HTTPS.

**6. Maintenance and Updates**

**System Performance Monitoring:** Monitoring of system performance against lag, bugs, or other security vulnerabilities constantly.

**System Updates:** Perform system updates in case of any kind of vulnerability discovered or addition of new features such as behavioral analysis or machine learning-based detection in future versions.

**7. Documentation and Reporting**

**User Documentation:** Clearly document the usage of the system, including how to upload files, view results, and generally navigate the platform.

**Developer Documentation:** This is to be technical documentation for system architecture, database schema, codebase, so that it is easily understandable for future developers contributing.

### ****Feasibility Study****

#### ****1. Technical Feasibility****

The project is technically feasible because it will be built using **Python (Django) and JavaScript**, both of which are well-supported and widely used for web applications. Key requirements include:

**Backend:** Django for handling file uploads, analysis, and storage.

**Frontend:** JavaScript for a user-friendly interface.

**Database:** PostgreSQL or SQLite for storing file analysis data.

**Security Measures:** File validation and safe storage to prevent malicious uploads from affecting the system.

#### ****2. Economic Feasibility****

The system will be developed using **open-source technologies**, meaning no licensing costs. Hosting the platform on a **local server or free-tier cloud services** (such AWS Free Tier) will keep expenses low. The project is financially viable for students, researchers, and small organizations with minimal infrastructure investment.

#### ****3. Operational Feasibility****

The system will have a **simple and intuitive user interface**, making it easy to upload files and view reports. It is designed for students, researchers, and cybersecurity enthusiasts who need a **basic malware detection tool** without complex configurations.

#### ****4. Time Feasibility****

The project is designed to be **completed within two months**, with clear milestones:

**Month 1:** Setting up the web interface, database, and file upload system.

**Month 2:** Implementing static analysis, classification, and report storage.

**Budget**

|  |  |
| --- | --- |
| **Development Tools and Software** | **Estimated Cost in ksh.** |
| SSL Certificate | 10,000.00 |
| Additional Security Plugins or Software | 5,000.00 |
| Development Machine (Laptop/PC) | 50,000.00 |
| Cloud Storage for File Uploads (beyond free tier) | 3,500.00 |
| Domain Name Registration. | 3,000.00 |
| Django Framework | 0.00 |
| Developer | 100,000.00 |
| **Total** | 171,500.00 |

**Detailed plan of the activities**

**Week 1: Requirement Analysis & Planning**

Define the scope of the work and objectives.

User requirement gathering: Identify the target users and their requirements regarding uploading files for analysis and report generation.

Existence of existing tools: Similar malware analysis systems are to be reviewed for identification of gaps and opportunities.

Finalize the project plan and set milestones.

**Week 2: System Design & Setup**

System Architecture: Design the overall system structure, which would be based on the client-server model, where Django would handle the backend and frontend.

Database Design: Design the schema of the database, which should contain tables for user data, uploaded files, and analysis results.

Frontend Design: Wire-framing and mockup for UI/UX.

**Week 3: Frontend Development**

Front-end development of interface: Put the skeleton with HTML, CSS, and JavaScript, providing basic functionalities to the user interface.

Designing the Interface for File Upload: Implementation of form for file upload.

Testing: Perform basic tests at the frontend regarding functionality.

**Week 4: Back-End Development**

Django Backend Setup: Setting up the Django project, configuration of models required for file analysis, and establishment of database connections.

Handling File Upload: Backend functionality to handle uploaded files, store them in the database, and provide security.

File Validation: File type and size validation to restrict malicious uploads.

**Week 5: File Analysis Implementation**

Static Analysis: Development of functionality that will analyze files using MD5, SHA-256 hash matching against known malicious signatures.

Integration of Backend with Frontend: The backend logic must be integrated with the frontend in terms of processing the file for analysis and presenting the results.

**Week 6: Report Generation & UI Refinement**

Generation of Analysis Reports: Give the facility to store the results of analysisładmetadata, hash values, threat classification-and show them.

UI Improvements: Improve the UI concerning the user experience and look-and-feel.

Testing: Perform user acceptance testing to make the interface user-friendly.

**Week 7: Security Implementation & Testing**

File Security: Introduce a file validation module, for example, scanning potentially malicious code within uploaded files.

Basic Access Control: Implement user authentication and access control.

Security Testing: Perform security testing, for instance, penetration testing to ensure that the system does not have vulnerabilities against malicious file uploads.

Bug Fixing and Improvement of Stability.

**Week 8: Final Testing & Deployment**

Final Testing: In this stage, comprehensive testing should be performed, including unit tests, integration tests, and security tests.

Deploy on Cloud: The application will be deployed on the cloud using AWS or Heroku.

Setup SSL and Hosting: Setting up SSL and hosting on the platform, optimization of server settings for performance.

**Week 9: Documentation & User Guide**

Documentation: Developer documentation on system architecture, database schema, and codebase.

User Guide: Create a simple user guide in order to help users through the file upload and results interpretation process.

Final Review: Review all parts of the project for correctness and make necessary adjustments before handover.

**Week 10: Final Presentation & Project Handover**

Final Review and Debugging: Review once more to ensure everything works as expected.

Project Report: Document the development process, results, and possibilities for future enhancement.

Present the Project: Showcase the project by demonstrating its functionality and features.

#### ****Conclusion****

The **Malware Analysis and Detection System** aims to provide an accessible and efficient solution for basic malware detection. By focusing on static analysis techniques, this project offers a practical tool for individuals, students, and researchers to analyze suspicious files and classify them based on predefined threat levels. The use of widely adopted technologies like Django, JavaScript, and PostgreSQL ensures a stable, scalable, and user-friendly platform.

The system will not only help in detecting known malware signatures but also provide a foundational platform for future security research and analysis. It fills a gap in the availability of lightweight, cost-effective malware analysis tools, particularly for those who cannot afford complex, commercial solutions.

**Example of the dashboard**

****