www.ijprems.com
editor@ijprems.com

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 05, Issue 04, April 2025, pp : 2862-2871

e-ISSN : 2583-1062
Impact Factor : 7.001

AI-DRIVEN DETECTION OF MALICIOUS MACROS IN OFFICE DOCUMENTS

Anand Raj M[1] , MS. Chithra Devi [2]
[1,2]Rathinam College Of Arts And Science, India.

ABSTRACT

A Python-based tool automates the detection of macro-embedded malware in Microsoft Office documents. It utilizes the olevba module from the oletools library to extract Visual Basic for Applications (VBA) code from macro-enabled files, including Word and Excel documents. The extracted code is then analyzed using the Mistral AI model via the Ollama API, which evaluates the code for any malicious patterns. If malware is detected, the tool automatically deletes the infected file to prevent potential threats. The tool features a user-friendly interface built using PySimpleGUI, allowing users to initiate system-wide scans, monitor progress, and view a detailed history of deleted files. This automation streamlines the detection and mitigation of macro-based malware, reducing the risk of exposure to malicious code. By integrating advanced analysis with the Mistral model and providing real-time protection, the tool ensures swift responses to malware threats.

1. INTRODUCTION

Macro-embedded malware in Microsoft Office documents poses a significant threat to organizations by exploiting the trust users place in seemingly legitimate files. To address this issue, the proposed Python-based tool automates the detection and mitigation of such malware, offering a proactive defense mechanism against these threats. The tool is built around the olevba module from the oletools library, which specializes in extracting Visual Basic for Applications (VBA) code from various macro-enabled files, including Word and Excel documents. VBA macros, often used for legitimate automation tasks, can also be exploited by attackers to embed malicious code that runs when a file is opened. Detecting and analyzing these macros is crucial for preventing malware from spreading and causing damage within an organization's network.

Once the VBA code is extracted, the tool leverages the power of artificial intelligence through the Mistral model via the Ollama API. This AI model is trained to identify patterns commonly associated with malware. By automating the analysis of macro code, the tool eliminates the need for manual inspection, which can be time-consuming and error-prone. The integration of AI also enables the tool to quickly and accurately detect malicious macros, significantly reducing the risk of undetected malware infiltrating a system. If the analysis detects malware, the tool immediately deletes the infected file, ensuring that no further damage is done.

The user interface, developed using PySimpleGUI, enhances accessibility and ease of use. It allows users to initiate system-wide scans with a single click, making it simple for both technical and non-technical users to deploy the tool. As the scan progresses, the interface displays a real-time progress bar, providing users with visibility into the scanning process. Additionally, a comprehensive history log is maintained, which records details of all deleted files, giving users a clear audit trail of the actions taken by the tool.

By automating both the detection and deletion of macro-based malware, this tool offers a streamlined and effective solution for organizations looking to fortify their cybersecurity defenses. The reliance on AI, particularly through the Mistral model, ensures that detection is not only fast but also adaptive, capable of identifying new and evolving threats.

OBJECTIVE OF THE PROJECT

To create an automated tool that detects and eliminates macro-embedded malware in Microsoft Office documents. The final result will be a Python-based application capable of scanning various macro-enabled documents, extracting and analyzing their Visual Basic for Applications (VBA) code for malicious patterns, and automatically deleting infected files to prevent further security risks.

By the end of the project, the following objectives will be achieved:

Automated Detection: The tool will automatically scan all macro-enabled documents for embedded macros, extract the VBA code using the olevba module, and evaluate it for malware using the Mistral AI model via the Ollama API.

Malware Mitigation: Upon detection of malicious code, the tool will immediately delete the infected file to stop the spread of malware or potential exploitation.

User-Friendly Interface: A graphical interface, built with PySimpleGUI, will allow users to easily initiate scans, view real-time progress, and track deleted files through a detailed history log.

Enhanced Security: By automating the process of malware detection and removal, the tool will minimize human error, reduce manual intervention, and significantly improve an organization's defense against macro-based malware threats.

SCOPE OF THE PROJECT

- The scope of this project encompasses the following key features and functionalities:
- **Automated Detection:** The tool automates the detection of macro-embedded malware in various Microsoft Office files, including Word, Excel, and PowerPoint documents.
- **VBA Code Extraction and Analysis:** It extracts and analyzes the VBA code using the olevba module and evaluates it for malicious patterns with the Mistral AI model via the Ollama API.
- **Automated Malware Mitigation:** Upon detecting any malicious code, the tool automatically deletes the infected files to prevent potential threats and reduce the risk of malware proliferation.
- **User-Friendly Interface:** The application provides a user-friendly interface built with PySimpleGUI, enabling users to initiate system-wide scans, monitor scanning progress, and interact seamlessly with the tool.
- **Detailed History Log:** It tracks deleted files and maintains a detailed history log for user review, allowing organizations to audit actions taken by the tool effectively.
- **Real-Time Feedback:** Users receive real-time feedback during the scanning and deletion processes, enhancing transparency and user experience.
- **Cross-Platform Compatibility:** The tool is designed for cross-platform compatibility, ensuring functionality on Windows, macOS, and Linux systems.
- **Enhanced Security:** By automating the detection and removal of macro-based malware, the tool significantly enhances security by reducing the risk of exposure to potentially harmful macros and minimizing human error.

2. LITERATUTRE SURVERY

Overview of Macro-Embedded Malware

Macro-embedded malware is a prevalent form of cyber threat that specifically targets Microsoft Office documents, including Word, Excel, and PowerPoint files. Macros are small scripts written in Visual Basic for Applications (VBA) that automate repetitive tasks, enhancing user productivity. However, the same functionality that makes macros beneficial also makes them susceptible to exploitation by cybercriminals. Attackers can embed malicious code within these macros, which can execute when a document is opened, leading to unauthorized actions such as downloading additional malware, executing harmful commands, stealing sensitive data, or even compromising entire systems.

The risk posed by macro-embedded malware has grown significantly, particularly in the context of phishing attacks where unsuspecting users may open seemingly legitimate documents. Once executed, these macros can establish backdoors for attackers, allowing them to maintain persistence within a compromised system. The ability of macro-based malware to bypass conventional security measures, such as antivirus software and firewalls, underscores the urgent need for more advanced detection and mitigation strategies.

Traditional Detection Methods

Traditional detection methods for macro-embedded malware primarily involve signature-based detection mechanisms implemented within antivirus solutions. These systems function by scanning documents for known malicious patterns, utilizing a database of previously identified malware signatures. While effective for recognized threats, signature- based methods are inherently limited; they often fail to detect new, modified, or previously unknown malware variants.

In addition to signature detection, manual inspection of VBA code remains a common practice among security analysts. However, this approach is fraught with challenges. Analyzing VBA code requires a high level of expertise to differentiate between benign and malicious macros, making it time-consuming and prone to human error. As attackers continually evolve their strategies, relying solely on traditional detection methods can leave organizations vulnerable to emerging threats.

Machine Learning in Malware Detection

The rise of machine learning has revolutionized the field of malware detection, particularly in addressing the limitations of traditional methods. Machine learning algorithms can analyze large datasets comprising known malware and benign code to identify patterns and characteristics typical of malicious macros. Unlike signature-based detection, machine learning is capable of generalizing beyond previously recognized threats, allowing it to detect novel or modified malware variants.

These algorithms can evaluate various features of documents, including file structure, code complexity, user behavior, and even the context in which macros are executed. By employing techniques such as supervised learning, unsupervised learning, and anomaly detection, machine learning models can classify documents based on their likelihood of containing malware. This adaptive capability enhances the effectiveness of malware detection, significantly reducing the window of opportunity for potential attacks.

AI-Driven Code Analysis

AI-driven code analysis has emerged as a sophisticated method for detecting macro- embedded malware, leveraging advanced techniques such as natural language processing (NLP) and deep learning. These systems analyze the underlying logic and structure of VBA code, going beyond surface-level signatures to assess the intent and functionality of macros. By understanding the contextual meaning of code constructs, AI models can identify subtle signs of malicious behavior that may elude traditional detection methods.

Integrating AI models, such as the Mistral model accessed via the Ollama API, enables real- time analysis of macro code. This capability allows for the rapid identification of malicious patterns as documents are opened or scanned, improving both the speed and accuracy of malware detection. Moreover, AI-driven analysis can continuously learn from new data, adapting to evolving threats and refining detection capabilities over time.

Recent Advancements and Tools

The field of macro-malware detection has seen significant advancements in recent years, particularly in terms of automation, real-time analysis, and enhanced detection techniques. Tools like olevba from the oletools library have been developed to facilitate the extraction and inspection of macros in Microsoft Office files. This tool simplifies the process of identifying potentially harmful code, enabling security analysts to focus their efforts on genuine threats.

Recent developments also include the integration of AI models and machine learning algorithms into detection tools, significantly improving the identification rates for novel and evasive malware variants. These advancements not only enhance detection accuracy but also allow for proactive measures against potential threats. Automated systems can monitor document access in real time, providing immediate feedback and responses to suspicious activities.

Furthermore, the trend towards combining multiple detection methodologies—such as signature-based, machine learning, and AI-driven analysis—into comprehensive security solutions allows organizations to leverage the strengths of each approach. This holistic view of malware detection equips organizations with robust strategies to neutralize threats before they can inflict significant harm, thus enhancing overall cybersecurity posture.

3. EXISTING SYSTEM

The detecting macro-embedded malware in Microsoft office documents is primarily managed through antivirus software and manual inspection tools. Antivirus programs scan files for known malware signatures and behaviors, but their effectiveness can be limited when dealing with new or unrecognized threats.

Security professionals often manually inspect macro-enabled files using tools like Microsoft's built-in features or third-party solutions to view and analyze the VBA code. While some enterprise solutions provide scanning capabilities for documents, they are generally not specialized in detecting malicious macros or fully automating the response.

These systems focus mainly on reactive detection, where threats are identified after they have been introduced, rather than proactively preventing their execution. This leaves some gaps in the ability to quickly and efficiently detect and mitigate risks associated with macro-based malware.

4. METHODOLOGY

System Architecture and Design

The Malware Detection Tool is designed with a modular architecture that integrates several components, ensuring efficient scanning and detection of macro-based malware in Microsoft Office documents. The architecture consists of three main layers:

System Architecture

The system architecture is designed to handle malware detection tasks efficiently by dividing the system into layers that manage different responsibilities.

User Interface Layer:

This layer is built using PySimpleGUI, providing an intuitive graphical interface for users to interact with the application. Key features of this interface include:

- **Buttons**: For initiating scans, viewing logs, and checking deletion history.
- **Progress Indicators**: Visual feedback on the scanning process and AI analysis.
- **Status Displays**: Real-time updates regarding ongoing operations and system performance.

Processing Layer:

The core functionality of the tool resides in this layer, responsible for:

www.ijprems.com
editor@ijprems.com

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 05, Issue 04, April 2025, pp : 2862-2871

e-ISSN : 2583-1062
Impact Factor : 7.001

- **Scanning the File System**: Detecting and identifying files that may contain macro-embedded malware.
- **Extracting VBA Code**: Utilizing libraries to pull macro code from Microsoft Office documents for analysis.
- **Interfacing with the Mistral AI Model**: This layer communicates with the Mistral model for analyzing the extracted code, enhancing detection capabilities through advanced AI techniques.
- **Thread Management**: This ensures the UI remains responsive during lengthy operations, improving the user experience.

Data Layer:

This layer manages the storage of deletion history, log messages, and application configurations. It employs in-memory data structures to track scanned files and their statuses, allowing for easy access during operations. Future enhancements may include integrating a database for persistent storage of logs and historical data.

User Interface Design

The user interface of the Malware Detection Tool is crafted to be user-friendly, catering to both technical and non-technical users. Key design features include:

- **Main Window**: Displays the tool's name, current operational status, and available actions, such as initiating a system-wide scan or viewing deletion history.
- **Progress Bars**: Two progress bars represent the scanning and AI analysis processes, providing users with visual feedback.
- **Log Output Area**: A multiline text area captures and displays logs that detail actions performed and any issues encountered during the scanning and analysis processes.
- **File Status Table**: A comprehensive table lists scanned files along with their statuses, enabling users to easily identify which files are pending, clean, or have been deleted.

Drive Detection and File Scanning

The tool employs a systematic approach to drive detection and file scanning. The following steps are executed upon initiating a scan:

Drive Detection: The application identifies all available drives on the system. For Windows systems, it utilizes the win32api library to retrieve logical drive strings. On Unix-based systems, the default is set to the root directory, ensuring compatibility across multiple platforms.

File Scanning: After detecting drives, the tool traverses each drive using os.walk(), identifying files with specific extensions associated with Microsoft Office documents (e.g., .doc, .docm, .dotm). For each detected file, the UI is updated to reflect the total number of files found and their current processing status.

Threading: The scanning process operates in a separate thread, preventing the GUI from freezing and allowing users to cancel the scan at any time, thus enhancing the overall user experience.

VBA Code Extraction

Extracting VBA code from Microsoft Office documents is crucial for effective malware detection. The application utilizes the oletools library's VBA_Parser class to facilitate this process:

- **VBA Detection**: The VBA_Parser first checks if the document contains any VBA macros. If macros are found, the tool proceeds to extract the relevant code.
- **Code Extraction**: The tool iterates through each detected macro, concatenating the VBA code chunks into a single string for subsequent analysis. This consolidated code is then prepared for AI evaluation.
- **Error Handling**: Any errors encountered during the extraction process are logged in the UI, providing users with insight into potential issues related to specific files.

AI-Based Analysis

The core of the malware detection process relies on an AI model, specifically the Mistral model, to analyze the extracted VBA code. The analysis process consists of the following steps:

- **Preparing the Prompt**: The tool constructs a prompt that incorporates the extracted VBA code along with specific instructions for the Mistral model to evaluate whether the code contains malware. This ensures the AI has a clear understanding of the task at hand.
- **API Request**: The application sends a POST request to the Mistral API with the prepared prompt, capturing the response that indicates whether malware was detected and providing an explanation of the findings.
- **Response Handling**: Based on the AI's response, the tool determines whether the detected code is malicious. If

malware is identified, the tool initiates the deletion process for the file; if not, it updates the file's status to indicate it is clean.

Malware Detection and File Deletion

Upon confirming that a file contains malicious VBA code, the tool executes immediate actions:

- **File Deletion**: The application utilizes the os.remove() function to delete the identified malicious file from the system, ensuring user protection from potential threats.
- **Logging Deletions**: Each deletion is meticulously logged, including the file path, reason for deletion, and a timestamp. This log is presented in the UI, allowing users to maintain a historical record of actions taken by the tool.
- **User Feedback**: The status of each scanned file is updated in the UI, enabling users to quickly identify which files were deleted due to malware detection and which were confirmed safe.

Logging and Reporting

Comprehensive logging and reporting mechanisms are integral to the Malware Detection Tool's design:

- **Log Storage**: The tool maintains an in-memory log that captures all significant actions, including file scans, deletions, and errors encountered during processing.
- **User Access**: Users can view logs directly within the application, facilitating quick reference and transparency regarding the actions undertaken by the tool.
- **Deletion History**: A dedicated feature allows users to view a comprehensive history of deleted files, including reasons for deletion and timestamps, providing insight into the tool's performance and effectiveness.

5. EXPERIMENTAL SETUP

System Overview

The malware detection system is designed to scan **Microsoft Office documents**, including **Word** and **Excel files**, for malicious VBA (Visual Basic for Applications) macros that can be used to execute harmful code. The system integrates advanced AI techniques for detecting malware, using the **Ollama API** with the **"mistral" model** for macro analysis. It is built as a Python-based tool that provides real-time feedback to users through a graphical user interface (GUI) built with **PySimpleGUI**. The key design principles of this system are usability, real-time malware detection, and automated actions like logging and file deletion.

This tool is particularly useful in environments that rely on Microsoft Office applications, where malicious macros embedded in files can pose significant security threats. By automatically scanning and analyzing Word and Excel documents, the system helps users and organizations protect themselves against potential malware attacks.

Key features of the system include:

- **File collection** from multiple storage locations.
- **Extraction of VBA macros** from Microsoft Word and Excel files.
- **AI-based malware analysis** through the Ollama API.
- **Real-time GUI feedback** showing file statuses and progress.
- **Automatic deletion** of malicious files and comprehensive logging for review.

The system targets users who may not have deep technical knowledge, offering a straightforward way to detect and mitigate risks posed by Office-based malware.

Environment

The environment setup for this malware detection tool is critical for ensuring compatibility and efficient performance. Below are the detailed system and software requirements necessary to run the tool effectively:

Operating System

The system is designed to run across different platforms and supports both **Windows** and **Unix- based operating systems (Linux/macOS)**. These operating systems are widely used in various environments, making the tool versatile and compatible with a broad range of hardware setups.

- **Windows**: The tool can run on Windows 7, 8, 10, and 11.
- **Linux**: Supported distributions include Ubuntu, Fedora, and other Unix-based variants.
- **macOS**: The tool is compatible with macOS systems (version 10.12 and higher).

Software Components

1. **Python 3.x**: The tool is written in Python, which is one of the most popular programming languages for malware

detection tools due to its simplicity, rich libraries, and cross-platform support. Python 3.x is required to ensure compatibility with the latest libraries and features used in the project.

2. **PySimpleGUI**: The GUI for the tool is built using PySimpleGUI, a Python library that simplifies the creation of user interfaces. It allows users to interact with the system, track scanning progress, and review malware detection statuses.

3. **Oletools (olevba)**: This open-source library is specifically designed for analyzing Microsoft Office files that contain embedded macros. The tool uses olevba to extract VBA macros from Word (.doc, .dotm, .docm) and Excel (.xls, .xlsx, .xlsm, .xlsb) files. It is highly efficient at recognizing embedded macros within these document formats, making it a key component of the system.

4. **Requests**: The requests library is a powerful Python module for making HTTP requests. It is used in this system to send extracted VBA macros to the **Ollama API** for further analysis and to retrieve the results of the malware scan.

5. **Ollama API (Local Instance)**: The tool uses the **Ollama API** for its AI-driven malware detection mechanism. The API leverages the **"mistral" model**, which is capable of analyzing VBA code to detect malicious behavior. For environments without internet access, the API can be deployed locally, ensuring secure and efficient analysis.

Hardware Requirements

For optimal performance, the following hardware specifications are recommended:

Minimum System Requirements:

RAM: 4 GB of RAM is needed to ensure that both the malware detection processes and the GUI run smoothly.

Processor: A dual-core processor is sufficient for moderate workloads.

Storage: 500 MB of available disk space is required for storing scanned files and log data.

Recommended System Requirements:

RAM: 8 GB or more to handle larger file volumes and concurrent scans.

Processor: A quad-core CPU to support faster processing times.

Storage: At least 2 GB of free storage for log files, temporary data, and potential API caching.

Network Requirements

- **Local instance of Ollama API**: If the Ollama API is installed locally, there is no need for internet access.
- **Cloud-based API instance**: When using the cloud version of the API, a stable internet connection is required to transmit the extracted VBA code for analysis and retrieve the results.

Data

The system processes Microsoft Office documents, particularly those containing VBA macros, which are analyzed for potential malware. It extracts VBA code from files and uses AI to assess malicious content. The tool logs analysis results, including file paths, timestamps, and actions taken, ensuring reliable detection and tracking of suspicious activity.

Input Data

The system scans **Microsoft Office files** stored on local drives or networked locations. These documents may contain VBA macros, which are the primary focus of the tool. Supported file formats include:

Microsoft Word:

- .doc
- .dotm
- .docm

Microsoft Excel:

- .xls
- .xlsx
- .xlsm
- .xlsb

These files are targeted because they have the potential to carry harmful macros, which are often used in phishing attacks or to deliver ransomware, steal sensitive information, or modify system settings maliciously.

Output Data

The tool provides the following outputs:

- **Malicious Files**: If the extracted VBA macros contain malware, the file is automatically deleted, and the corresponding log is updated with details about the action taken.

- **Clean Files**: If no malware is detected, the file is labeled as clean, and no further action is taken.
- **Log Files**: All actions taken by the tool are logged, including file names, file paths, timestamps, and analysis results.

Workflow

File Collection

- The first step of the workflow involves scanning directories or the entire file system for
- **Microsoft Word** and **Excel** files.
- The system uses Python's file-handling modules (os, glob) to recursively search through storage drives for any files that match the target extensions: .doc, .dotm, .docm,
- .xls, .xlsx, .xlsm, and .xlsb.
- Once a file is located, it is added to a queue for further processing.

Macro Extraction

- Once a file is identified, **olevba** is used to extract any VBA macros embedded within the file. This extraction process works across both Word and Excel file formats, identifying any potentially harmful macros hidden within.
- If the file contains macros, the system proceeds with further analysis; otherwise, the file is marked as clean and skipped for analysis.

AI-Based Malware Detection

- The extracted VBA macro is sent to the **Ollama API** (either local or cloud-based) for analysis. The API uses machine learning models (in this case, the **"mistral" model**) to assess the macro code and determine whether it exhibits malicious behavior.
- The API responds with a classification:

Malicious: The file is flagged as containing harmful code.

Clean: The file is considered safe, and no further action is needed.

- Files flagged as malicious are automatically deleted from the system to prevent potential harm.

Real-Time Progress Feedback

- Users can track the scan's progress through the PySimpleGUI interface:

A **file scan progress bar** updates as the system searches for and processes files.

A separate **analysis progress bar** shows the status of the AI-driven macro analysis.

- Each file's status (Pending, Clean, Malicious) is displayed in a **status table** in the GUI.

Logging and Reporting

- The system maintains detailed logs of every action performed:

Files that were scanned and their results.

Files that were deleted for containing malicious code.

Errors encountered during the scanning process.

- Logs are stored locally and can be exported for further auditing or reporting purposes.

User Interface

The user interface, developed using **PySimpleGUI**, is designed to be user-friendly and intuitive, even for non-technical users.

Start/Stop Scans

- The GUI includes simple buttons for starting and stopping the scan process.
- Upon clicking "Start," the system begins scanning the file system for Word and Excel files containing VBA macros.

File Status Table

- A table in the GUI displays the current status of each file:

Pending: The file is in the queue for scanning.

Clean: The file has been analyzed and found to be safe.

Malicious: The file contains harmful macros and has been deleted.

Progress Bars

Two progress bars provide real-time feedback:

- Detection progress: Tracks the system's progress as it searches the file system.
- File Analysis progress: Shows the status of each file being analyzed by the Ollama API

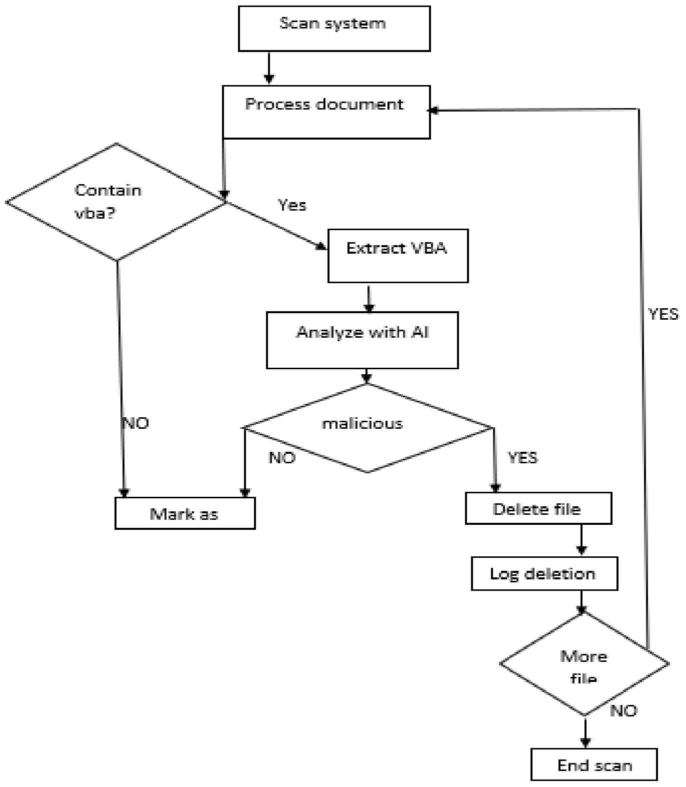
Error Messages

Any errors encountered during file scanning or macro extraction are displayed in the GUI. These include issues like permission errors, file corruption, or missing macros.

Logs

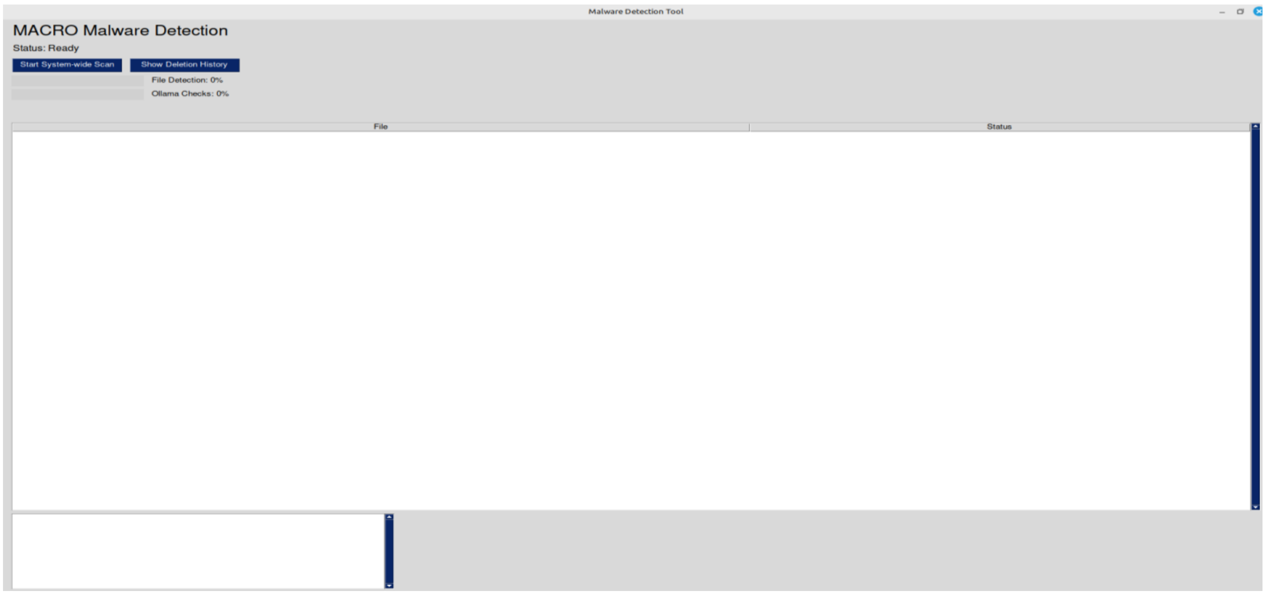
Users can export logs directly from the interface for further investigation or record-keeping.

Flow diagram



Sample output

Scanning document


Malware Detection Tool

MACRO Malware Detection

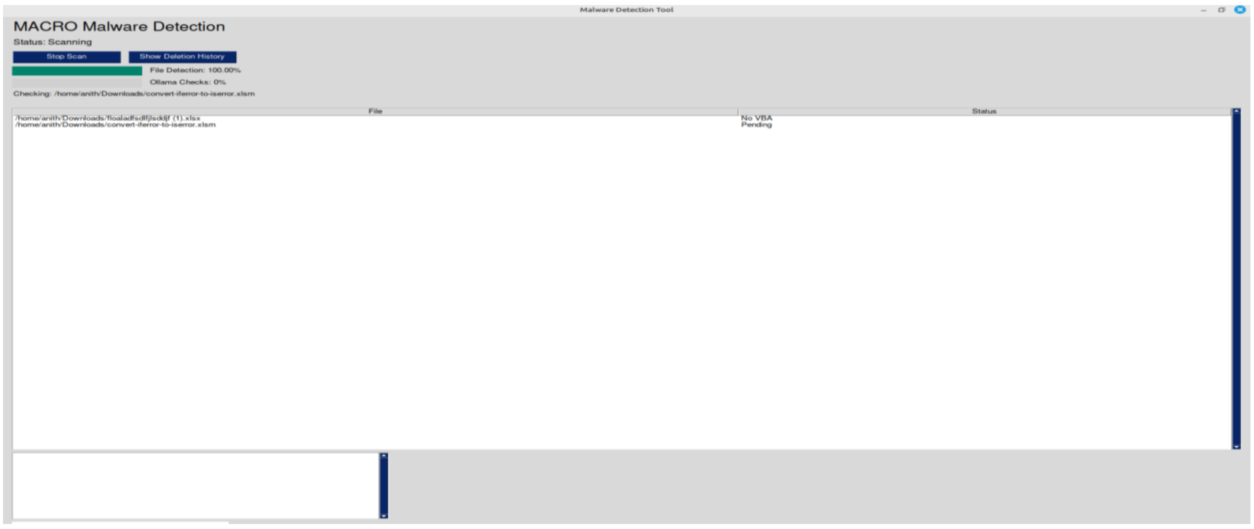
Status: Ready

File Detection: 0%

Ollama Checks: 0%

File	Status

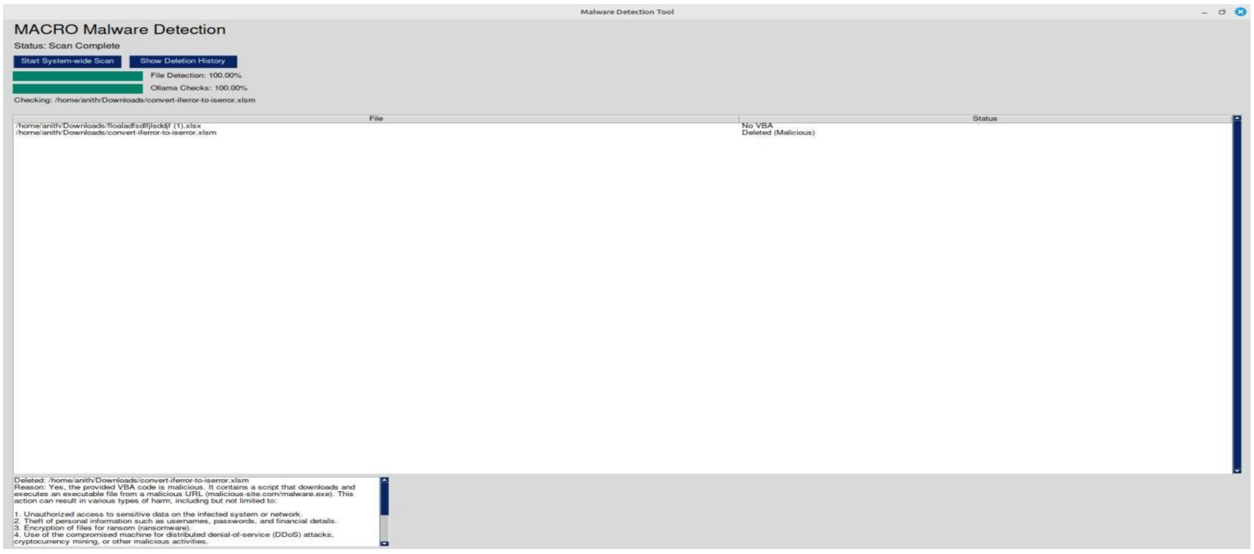
Figure Scanning the file



```
Malware Detection Tool
MACRO Malware Detection
Status: Scanning
[Stop Scan] [Show Deletion History]
File Detection: 100.00%
Ollama Checks: 0%
Checking: /home/anith/Downloads/convert-iferror-to-iserror.xlsm

File                                                   Status
/home/anith/Downloads/floaladfsdlfjlsddjf (1).xlsx     No VBA
/home/anith/Downloads/convert-iferror-to-iserror.xlsm  Pending
```

Figure Scanning the file

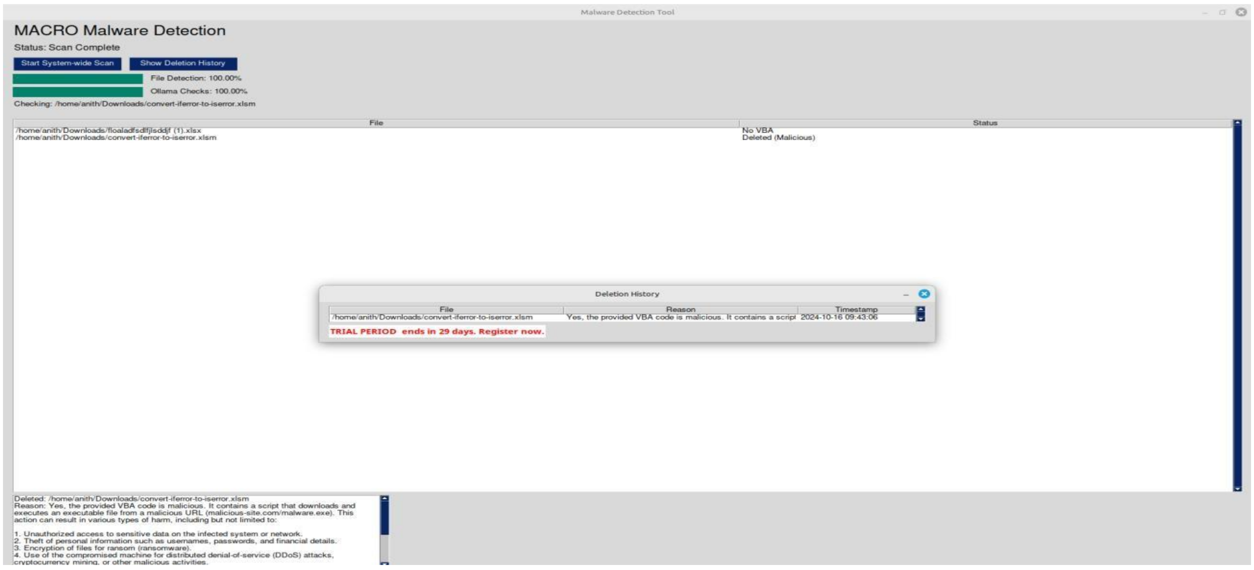
Report of the scanned document



```
Malware Detection Tool
MACRO Malware Detection
Status: Scan Complete
[Start System-wide Scan] [Show Deletion History]
File Detection: 100.00%
Ollama Checks: 100.00%
Checking: /home/anith/Downloads/convert-iferror-to-iserror.xlsm

File                                                   Status
/home/anith/Downloads/floaladfsdlfjlsddjf (1).xlsx     No VBA
/home/anith/Downloads/convert-iferror-to-iserror.xlsm  Deleted (Malicious)

Deleted: /home/anith/Downloads/convert-iferror-to-iserror.xlsm
Reason: Yes, the provided VBA code is malicious. It contains a script that downloads and
executes an executable file from a malicious URL (malicious-site.com/malware.exe). This
action can result in various types of harm, including but not limited to:

1. Unauthorized access to sensitive data on the infected system or network.
2. Theft of personal information such as usernames, passwords, and financial details.
3. Encryption of files for ransom (ransomware).
4. Use of the compromised machine for distributed denial-of-service (DDoS) attacks,
cryptocurrency mining, or other malicious activities.
```

Figure Report of scanning file

Log of deletion of document



```
Malware Detection Tool
MACRO Malware Detection
Status: Scan Complete
[Start System-wide Scan] [Show Deletion History]
File Detection: 100.00%
Ollama Checks: 100.00%
Checking: /home/anith/Downloads/convert-iferror-to-iserror.xlsm

File                                                   Status
/home/anith/Downloads/floaladfsdlfjlsddjf (1).xlsx     No VBA
/home/anith/Downloads/convert-iferror-to-iserror.xlsm  Deleted (Malicious)

Deletion History
File                                                   Reason                                                        Timestamp
/home/anith/Downloads/convert-iferror-to-iserror.xlsm  Yes, the provided VBA code is malicious. It contains a script  2024-10-16 09:43:06
TRIAL PERIOD  ends in 29 days. Register now.

Deleted: /home/anith/Downloads/convert-iferror-to-iserror.xlsm
Reason: Yes, the provided VBA code is malicious. It contains a script that downloads and
executes an executable file from a malicious URL (malicious-site.com/malware.exe). This
action can result in various types of harm, including but not limited to:

1. Unauthorized access to sensitive data on the infected system or network.
2. Theft of personal information such as usernames, passwords, and financial details.
3. Encryption of files for ransom (ransomware).
4. Use of the compromised machine for distributed denial-of-service (DDoS) attacks,
cryptocurrency mining, or other malicious activities.
```

Figure Log of deletion of file

6. RESULTS AND DISCUSSIONS

The Malware Detection Tool, utilizing AI-driven techniques, yielded promising results and initiated valuable discussions.

The tool effectively identified macro-embedded malware within Microsoft Office documents, showcasing a high degree of accuracy in analyzing extracted VBA code through the Mistral AI model. The implementation of this detection mechanism significantly reduced the risk of malware spread by promptly deleting infected files.

Discussions surrounding the project highlighted key operational considerations, such as the efficient extraction of VBA code and the seamless integration of the user interface. The importance of continuous improvement was underscored, emphasizing the need for regular updates to the AI model in response to emerging threats and changing cybersecurity landscapes. Overall, the Malware Detection Tool demonstrated the effectiveness of AI in enhancing security measures, addressing macro-based malware threats, and providing critical insights into malware detection processes.

7. CONCLUSION AND FUTURE ENHANCEMENT

Conclusion

The Malware Detection Tool has proven to be an effective solution for identifying and mitigating macro-embedded malware within Microsoft Office documents. By leveraging advanced AI techniques and a user-friendly interface, the tool successfully streamlines the detection process, enhances security measures, and reduces the risk of malware spread. The implementation of the Mistral AI model significantly improves the accuracy of malware identification, providing organizations with a robust defense against evolving threats. As cybersecurity challenges continue to grow, this tool represents a crucial step toward safeguarding sensitive information and maintaining a secure operational environment. Future enhancements will focus on integrating more advanced AI models and expanding the tool's capabilities to ensure ongoing protection against emerging malware threats.

Future Enhancement

The Malware Detection Tool presents several opportunities for future enhancements and expansions. One key area for development is the integration of additional AI models, which could improve detection accuracy and broaden the tool's capabilities to identify various types of malware beyond macro-embedded threats. Transitioning from in-memory log storage to a dedicated database would enhance the tool's ability to manage historical data, allowing for more extensive analysis and reporting. Furthermore, implementing user customization features would enable users to adjust scanning parameters and notification settings, thereby improving the overall user experience and adaptability to specific organizational needs.Additionally, further refining the tool for consistent performance across different operating systems, including mobile platforms, could broaden its accessibility. Establishing a mechanism for regular updates and integrating threat intelligence feeds would ensure that the tool remains current with emerging malware trends and evolving attack vectors.

8. REFERENCES

[1] Alazab, M., & Weddell, G. (2019). **Macro Malware: A Comprehensive Survey of the Evolution and Mitigation Techniques**. ACM Computing Surveys, 52(6), 1-38. DOI: 10.1145/3297338.

[2] Symantec Corporation. (2020). **Internet Security Threat Report**. Retrieved from https://www.broadcom.com/company/newsroom/press-releases?filtr=2020-04-21-04-00

[3] Shafique, M., Alzahrani, H., & Khan, M. K. (2021). **A Machine Learning Approach for Malware Detection in Office Documents**. IEEE Access, 9, 14510-14523. DOI: 10.1109/ACCESS.2021.3059287.

[4] Bansal, A., & Kumar, A. (2020). **Analyzing Macro-Embedded Malware through Machine Learning Techniques**. International Journal of Computer Applications, 975, 1-6. DOI: 10.5120/21901-7603.

[5] Brumley, D., & Neis, P. (2019). **A Survey of Malware Detection Techniques**. IEEE Security & Privacy, 17(2), 10-22. DOI: 10.1109/MSP.2019.2896552.

[6] Alshahrani, A., & Alzahrani, A. (2021). **Utilizing Machine Learning for Malware Detection in Office Files**. Journal of Information Security and Applications, 57, 102680. DOI: 10.1016/j.jisa.2020.102680.

[7] Li, Z., & Wang, H. (2020). **Machine Learning for Malware Detection: A Review**. Journal of Network and Computer Applications, 169, 102756. DOI: 10.1016/j.jnca.2020.102756.

[8] Rollings, A., & Hu, X. (2022). **Towards Automated Malware Detection: Using Machine Learning for Analyzing VBA Macros**. Journal of Cybersecurity and Privacy, 2(1), 145-162. DOI: 10.3390/jcp2010010.

[9] OLETools Documentation. (2024). **oletools: Analyzing OLE Files with Python**. Retrieved from https://github.com/decalage2/oletools

[10] Bace, R. G., & Landwehr, C. E. (2003). **Computer Security Handbook**. Wile