Malware Detector

Final Report

Information Warfare - IE4032

IT20258658

Abeywickrama G.D.C.J

Contents

List of figures

Introduction

With computers and other connected devices playing such a big part in our daily lives in the digital age, the threat of malicious software, or malware, is more real than ever. Malware can take many different forms, ranging from sophisticated ransomware that can encrypt your files to viruses that can corrupt your data. In addition to endangering people's security and privacy, it also puts governments, businesses, and the world economy at serious risk. Malware is always evolving, which makes it difficult to stop since hackers are always coming up with new ways to get around even the strongest security measures. Developing proactive and efficient countermeasures is imperative considering the constantly changing threat landscape. Our project introduces a user-friendly malware detection tool in response to this pressing need.

The purpose of the Malware Detection Tool is to empower people and organizations by offering a simple yet effective solution to the ever-growing threat of malware. We understand that a tool that is accessible and user-friendly is necessary for the wide spectrum of computer users, from tech-savvy experts to those who are less familiar with cybersecurity. This requirement is met by our tool, which provides an easy-to-use graphical interface that makes it easier to scan files for possible malware. The fundamental function of the Malware Detection Tool is its ability to detect and identify malware by utilizing machine learning, a branch of artificial intelligence. It makes intelligent predictions regarding the existence of malware in files by utilizing a Random Forest Classifier, a pre-trained machine learning model.

Using an extensive amount of data, this model is trained to extract features from Portable Executable (PE) files, which are frequently present in Windows programs and applications. The PE file format includes details in its header that may indicate malicious intent, along with other important information about how a program should operate. To determine whether malware may be present, our tool extracts these characteristics from the PE files and runs them through a machine learning model. The model raises a flag to notify the user of a possible threat if it finds traits linked to malware.

The Malware Detection Tool is thoroughly examined in this project report, which also discloses the tool's inner workings, methodology, and outcomes. It explores the design of the user interface, enabling users to interact with the tool and comprehend how it works with ease. The underlying machine learning model and its function in malware detection are covered.

The report also offers insights into the evaluation's findings, demonstrating the efficacy and accuracy of our model. The Malware Detection Tool has advantages, but it also has drawbacks and room for improvement. The report discusses possible improvements considering the constantly changing threat landscape. These include implementing sophisticated machine learning methods and broadening the range of datasets to achieve even higher levels of accuracy and precision. We hope that this report's insights will give you a better understanding of the Malware Detection Tool and its role in protecting the digital world as we set out on this journey through the world of malware detection. There has never been a more pressing need for creative solutions like this tool, especially with malware as a constant threat.

Overview of the product

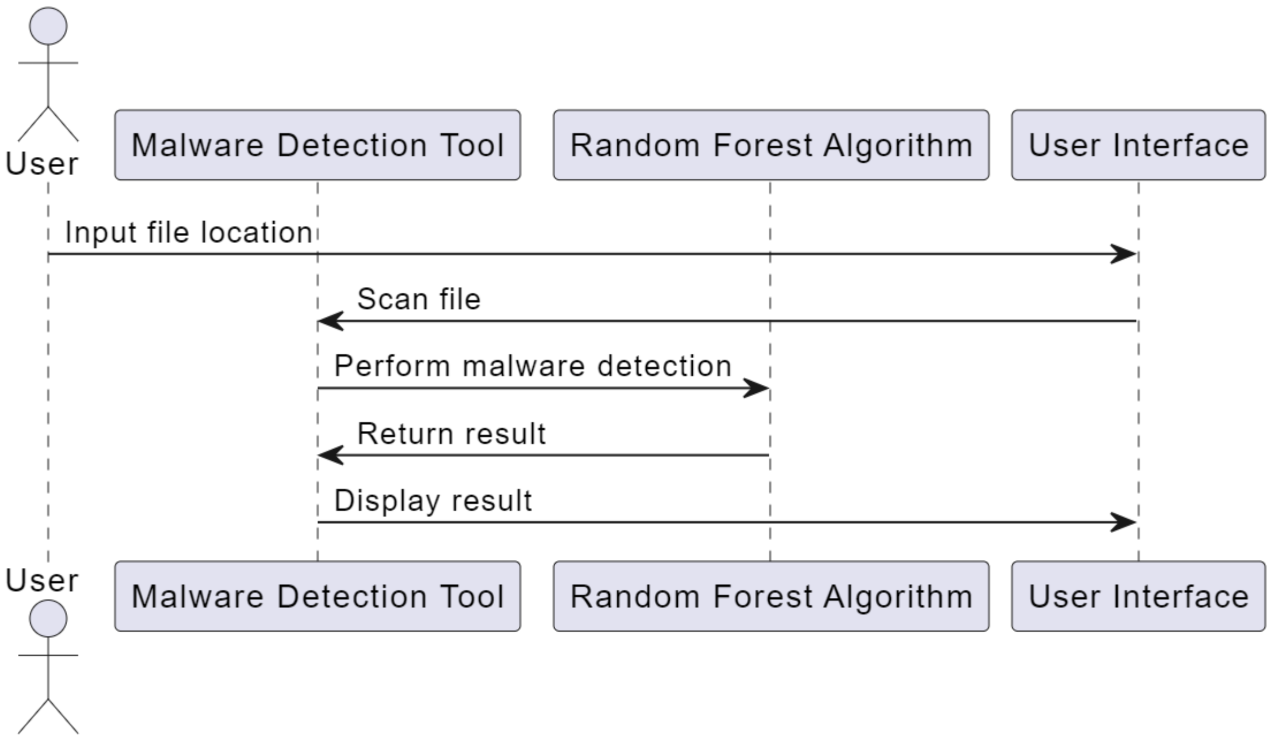


Figure 1 Overview of the product

PE files

The term "PE headers" may seem like a mysterious acronym in the world of computer systems, but it's essential to comprehending how software works. Executable files' digital identity cards are called PE headers, or Portable Executable file headers. Software programs rely on PE headers to transmit vital information about their structure and execution requirements, just as we carry identification to verify our identity. The Portable Executable (PE) file format, which is widely used in Windows operating systems, is most frequently linked to these headers. These are the first few bits of information that a computer sees when it loads an executable file. PE headers hold important information about the file, including its size, the program type, where certain sections start, and much more.

PE headers are essentially the "user manual" for a program on a computer. They provide instructions to the computer on how to install, launch, and use the application. The operating system needs this data to guarantee that the program runs properly. This are the main method how we identify malware or not in our tool.

The Random Forest Algorithm in Malware Detection

Within the complex field of malware detection, the Random Forest algorithm stands out as an example of an advanced data classification and analysis system. It is a suitable option for our malware detection tool because it finds its value in the complex world of malware for a variety of reasons. Random Forest is a group of decision trees, not just another algorithm. It is an expert at classification, like a great detective who looks through a wide range of clues to draw a meaningful conclusion. Random Forest's strength is its ability to handle complex data, which allows it to adapt very well to the varied and constantly changing malware landscape. One essential feature of Random Forest that fits well with malware's ever-changing nature is its adaptability. Its ability to recognize complex patterns and anomalies in data is essential for detecting malware, which frequently uses complex obfuscation techniques to hide its true purpose. This algorithm finds the subtleties that separate potentially dangerous files from benign ones, acting as a kind of digital sleuth. It's also important to note Random Forest's scalability. Its ability to handle large datasets is essential in the big data era, as there is a plethora of potential threats. This scalability guarantees that our tool will work effectively when examining a wide variety of files, from single program to large software suites.

The Random Forest ensemble's collective intelligence reduces the possibility of overfitting, a condition in which a model becomes unduly specialized and loses its capacity for generalization. This is a crucial feature for malware detection, where a strong model needs to reliably detect threats across a range of variations while avoiding overfitting traps. Random Forest serves as the cornerstone of our machine learning model in our malware detection tool. It carefully processes the feature vectors that are extracted from the PE headers after receiving them. Like a keen detective, the machine learning model trained with Random Forest recognizes the telltale signs of possible malware.

Relationship PE file and Random Forest

PE headers and our tool's Random Forest algorithm have a mutually supportive relationship. The first filter is the PE header, which directs our tool's attention to files with suspicious header characteristics. By doing this, the number of false positives is decreased, and the tool's scrutiny is focused on the most important areas. Conversely, the Random Forest algorithm improves the precision and breadth of our analysis. It carefully analyses the data that is taken from PE headers, identifying even the smallest patterns that could point to malware.

PE headers serve as the foundation for this dynamic combination, and the Random Forest algorithm enhances the detection process. As a result, a reliable and efficient malware detection system that can detect possible threats and reduce false alarms has been created. This relationship is a prime example of how technical expertise and careful analysis can be combined in our continuous pursuit of digital security.

Scope

Information security in the context of our product covers a wide range of fields, distinguished by the complexity of technical details as well as the critical need of protecting digital environments. Fundamentally, this scope includes a thorough examination of Portable Executable (PE) file headers, led by the accuracy of the Random Forest algorithm, with the unwavering goal of locating and minimizing possible malware threats.

This scope covers a wide range of topics, including anomaly detection, digital fingerprints, and data structures. It interacts with the constantly changing malware landscape, constantly adjusting to the ever-changing strategies of cyber adversaries. Moreover, the purview encompasses the coordination of sophisticated methods in predictive modelling, machine learning, and data analysis. Here, we make use of the Random Forest algorithm's incredible power, which helps our tool identify even the smallest deviations and anomalies in file structures. This complex analysis is the cornerstone of proactive defense, acting as a sentinel to prevent cyberattacks. The information security strategy used by our product is essentially an ongoing search for digital fortification. It navigates the terrains of PE file structures, interpreting their complexities, and it collaborates with machine learning to become an attentive defender. The breadth, demonstrated by the accuracy and depth of our analysis, reflects the constantly growing field of digital security. The breadth of information security in our product is evidence of our unwavering dedication to protecting the digital world, one PE header at a time, in a time when cyber threats are ever-changing.

Methodology used to develop.

Our malware detection technology is the result of a carefully planned procedure that incorporates a wide range of approaches and strategies to guarantee its reliability and effectiveness. This section lists the main techniques used in the development stage.

This tool based on the machine learning algorithm. In the machine learning we must do various implementation; these are the step-by-step process we achieved during the tool development.

Dataset preparation

- **Dataset Collection**
  My tool is more based on the machine learning, and data collection is very important. During the development stage I gather data set from Kaggle.com web site **[1]** I have mentioned the data set in the reference section.

- **Data Preprocessing**
  After selecting data set, I recreated the data set and remove the duplications and other null values.

- **Annotation and Labeling**
  After the data preprocessing stage, I changed labeled of the data set for compatible with the feature extraction part.

- **Dataset split.**
  Subsets of the dataset were created for testing, validation, and training. This division made it possible to train the machine learning model on one subset, check its performance on another, and then test it on an additional, unknown dataset to see how well it performed in the real world.

Feature Extraction

Feature extraction from Portable Executable (PE) file headers is a fundamental procedure in the field of malware detection. To help our machine learning model determine the type of executable files, it must extract and process important characteristics from these headers. This is a thorough examination of this crucial stage.

The process of extracting features starts with a methodical approach to accessing and analyzing the PE headers. **'e_magic,' 'e_cblp,' 'SizeOfCode,' 'SizeOfImage,' 'SizeOfHeaders,' 'SizeOfInitializedData,' 'SizeOfUninitializedData,' 'SizeOfStackReserve,' 'SizeOfHeapReserve,' 'NumberOfSymbols,' and 'SectionMaxChar.'**

1. **'e_magic' - The Magic number:**
   The "magic number" indicating the format of the file is stored in the 'e_magic' field, which is the first field in the headers. Our model determines the file's structure and compatibility by decoding this number, providing a critical understanding of the detection process.

2. **Size Is Important:**
   Entries such as 'SizeOfCode,' 'SizeOfImage,' and 'SizeOfHeaders' provide information about the dimensions and arrangement of the executable file. The size of the code section is indicated by the 'SizeOfCode', and the overall size of the image when loaded into memory is revealed by the 'SizeOfImage'. The size of the headers themselves is specified by "SizeOfHeaders."

3. **Data Segmentation:**
   The variables 'SizeOfInitializedData' and 'SizeOfUninitializedData' offer information regarding the dimensions of data segments, emphasizing regions that might contain inconsistencies or irregularities.

4. **Memory Reservations:**
   Features such as 'SizeOfStackReserve' and 'SizeOfHeapReserve' highlight memory allotments, which are essential to comprehending the behavior and resource needs of the program.

5. **"NumberOfSymbols" –**

Overview of the Code: The number of symbols in the executable file is related to the 'NumberOfSymbols' attribute, which helps the model understand the complexity and organization of the code.

6. **'SectionMaxChar' –**

Characterizing Sections: 'SectionMaxChar' concentrates on the features of file sections, recognizing crucial characteristics that distinguish them and might disclose malignant patterning.

Model Selection

A crucial choice in the creation of our malware detection tool is choosing the right machine learning model. The tool's ability to distinguish between malicious and benign software is powered by the selected model. Here's a closer look at the model selection procedure.

- **The Model Selection Landscape:**

  There is a wide variety of algorithms in the field of machine learning, each one designed to handle tasks. The decision for malware detection came down to an algorithm that could handle complicated data and adjust to the constantly changing nature of malware. Within this framework, Random Forest showed itself to be a strong competitor.

- **Why Random Forest:**

  Random Forest is an adaptable method for group learning. Its capacity to generate a collection of decision trees, each contributing its own insights and forecasts, is what makes it unique. Random Forest increases the accuracy of the model and reduces the possibility of overfitting by combining these individual tree predictions. This flexibility is crucial in the field of malware detection since the threat landscape is always changing.

Tool development

During the tool development, I used pickle format to integrate trained algorithm to the tool. Pickle is format acts as a small archive that makes it simple for machine learning practitioners to serialize, deserialize, and store their models.

Here are the used python libraries for development,

- **PyQt5**

  This is framework used to develop the user interface (Graphical user interface (GUI)-based desktop applications)

- **scikit-learn.**

  A potent Python machine learning library is called Scikit-learn. Numerous tools for classification, regression, clustering, and other tasks are available. To implement and train the Random Forest machine learning model for malware detection, your code makes use of scikit-learn.

- **Pefile**

  The Python module pefile offers reading and manipulating capabilities for Portable Executable (PE) files, which are frequently utilized in Windows applications. Information is extracted from PE file headers using it.

- **pandas**

  Pandas is a well-liked Python package for data analysis and manipulation. It offers functions and data structures for handling structured data, including tables and data frames. Pandas is used in your code to handle and process data, particularly in the malware detection procedure.

- **Seaborn**

  A data visualization library called Seaborn was constructed atop Matplotlib. It offers a sophisticated user interface for producing visually appealing and educational statistical graphics. Seaborn is used in your code to generate distribution and count plots, among other visualizations.

- **matplotlib.**

  Matplotlib is a flexible Python data visualization library. It makes it possible to create a huge range of plots and charts that are static, animated, or interactive. Matplotlib is used in conjunction with Seaborn in your code to visualize data.

These are the approaches used in the tool's development. This section includes detailed instructions on how to choose the model, prepare the data, choose the dataset, and integrate the trained model with the tool. The PyQt5 library was also used in the development of the user interface and pefile used for extract inputted files PE headers.

Methodology that the product work

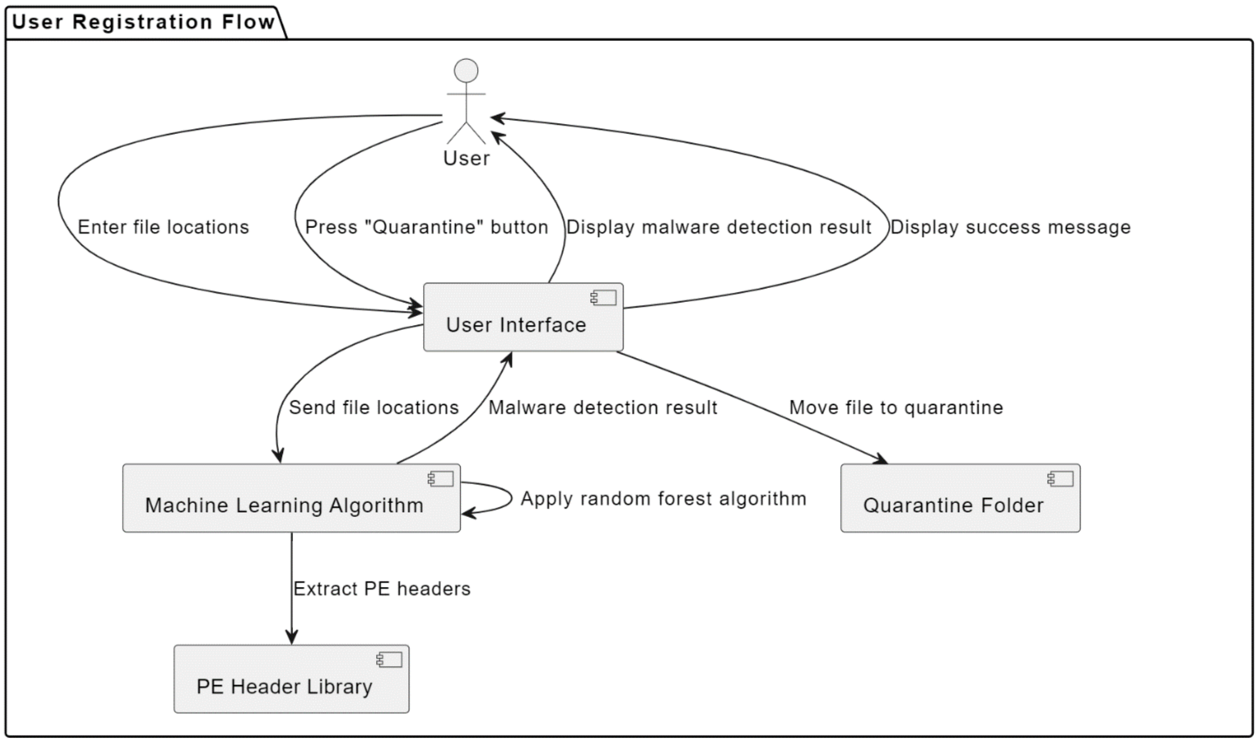


Figure 2 System Diagram

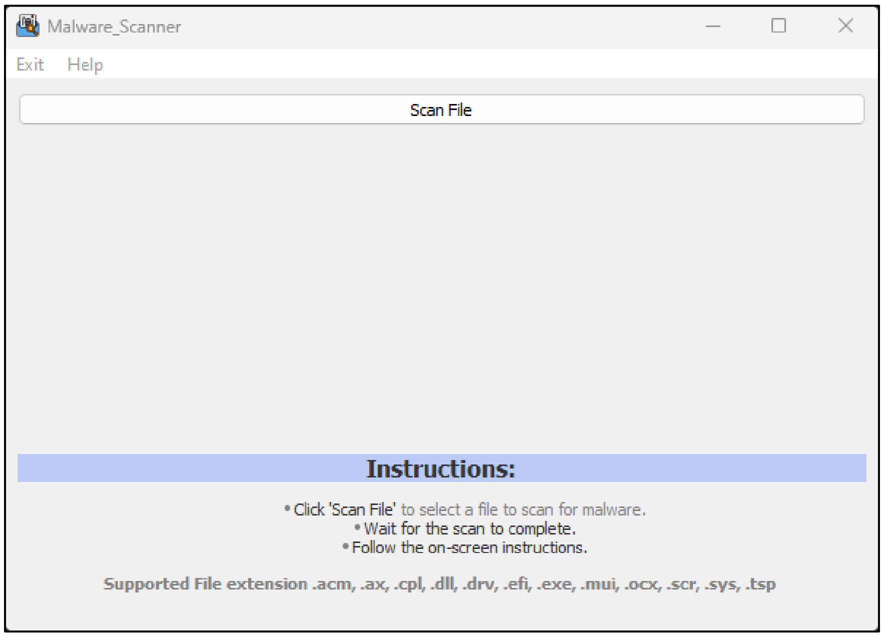
The tool operates according to a simple procedure. Initially, the user enters the file path of a potentially harmful file into the interface of our tool. After that, the tool forwards this file to a Python script we wrote to examine each file's PE headers. The pefile Python library, which extracts all pertinent data from the file's PE header, is used to perform this analysis. After that, the data is arranged into variables for additional processing.

After that, an algorithm that has been trained receives this data. The purpose of this algorithm is to examine the PE header and identify any malicious code present in the file. The algorithm outputs a "1," signifying the presence of malware if it finds any. It outputs a "0," indicating that the file looks safe if malware is not found.

Our tool notifies the user right away if malware is found, letting them know that they could be in danger. Quarantining the file is an option that is offered in the message. The file is moved to a special quarantine folder and has its file extension changed to ". malware" for easy identification if the user decides to quarantine it. On the other hand, the tool gives reassurance with a message that says, "Malware not detected; you are safe," if the algorithm finds no evidence of malware.

Here is the simple way of the implementation of the tool.

User Interface

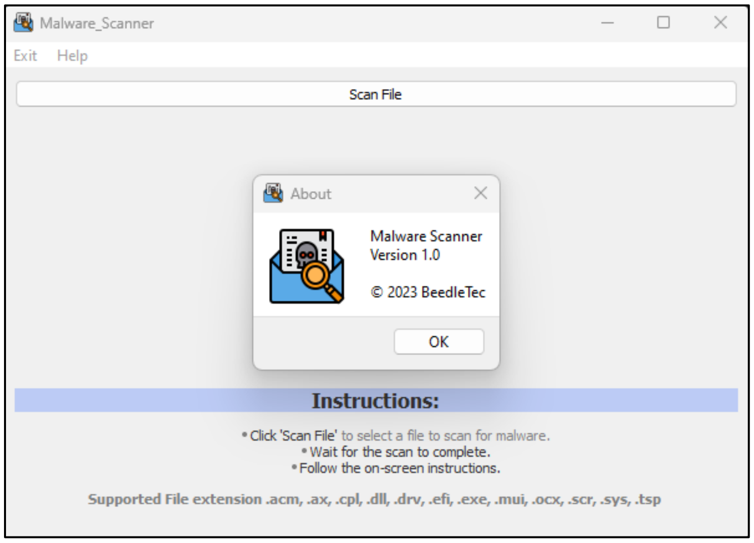


Instructions:

• Click 'Scan File' to select a file to scan for malware.
• Wait for the scan to complete.
• Follow the on-screen instructions.

Supported File extension .acm, .ax, .cpl, .dll, .drv, .efi, .exe, .mui, .ocx, .scr, .sys, .tsp

Figure 3 User Interface 1



Malware Scanner
Version 1.0

© 2023 BeedleTec

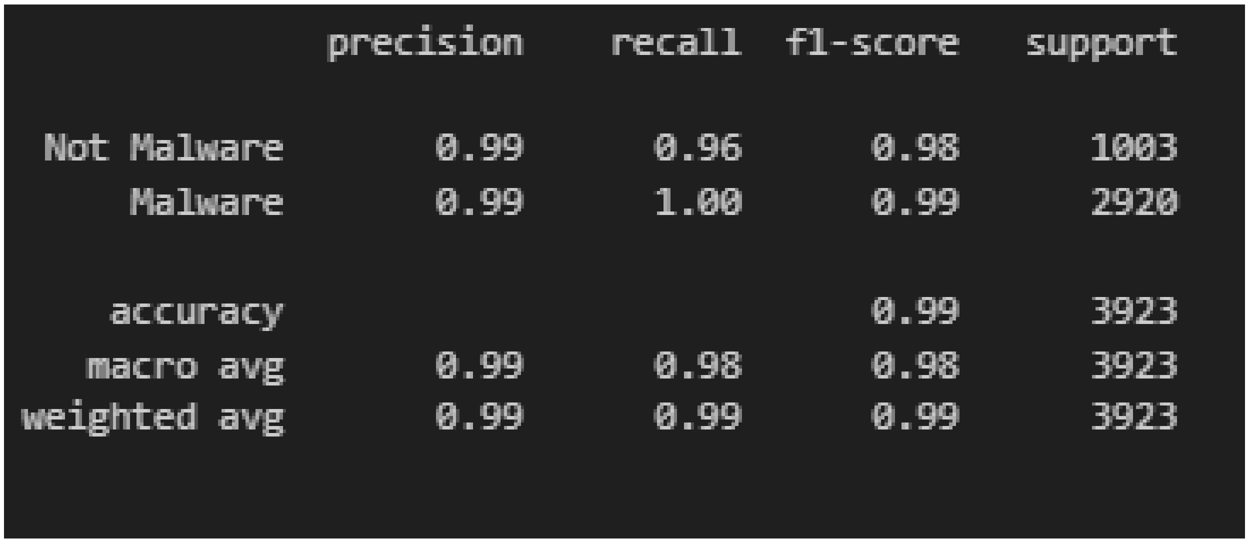
Instructions:

• Click 'Scan File' to select a file to scan for malware.
• Wait for the scan to complete.
• Follow the on-screen instructions.

Supported File extension .acm, .ax, .cpl, .dll, .drv, .efi, .exe, .mui, .ocx, .scr, .sys, .tsp

Figure 4 User Interface 2

Accuracy of the machine learning model



```
              precision    recall  f1-score   support

 Not Malware       0.99      0.96      0.98      1003
     Malware       0.99      1.00      0.99      2920

    accuracy                           0.99      3923
   macro avg       0.99      0.98      0.98      3923
weighted avg       0.99      0.99      0.99      3923
```

Figure 5 Accuracy

Conclusion

To sum up, the creation of this malware detection tool is the result of the fusion of data analysis, user-centric design, and state-of-the-art technology. We have developed a strong system that can quickly and accurately identify potentially harmful files by utilizing machine learning. The technical prowess of the tool is demonstrated by its ability to analyze PE headers and make well-informed decisions regarding the presence of malware. Machine learning models, in particular the Random Forest algorithm, are used to demonstrate the tool's dedication to accuracy and effectiveness. It demonstrates our commitment to remaining ahead of the constantly changing world of cyberthreats. Furthermore, even non-technical users will be able to easily navigate the tool thanks to its user-friendly interface, which was built with the PyQt5 library. It does a good job of bridging the gap between complex backend operations and an intuitive front end. The importance of tools such as these is immeasurable, given the ongoing deluge of malicious software into the digital sphere. With its real-time analysis, our tool gives users the ability to protect their systems by enabling them to make informed decisions about potentially dangerous files.

Reference

1. Dataset - https://www.kaggle.com/datasets/amauricio/pe-files-malwares/data
2. Google Drive (tool & Video)
   https://drive.google.com/drive/folders/1jhUDqqS_9oGaIIHp_ppx_iV2IRf_Xl_Q?usp=sharing