**FINAL REPORT**

For

**MALWARE DETECTION USING ML and Python**

Prepared by

|  |  |  |
| --- | --- | --- |
| **Specialization** | **SAP ID** | **Name** |
| B.Tech LLB-Cyber Law | 500068884 | Isha Mittal |
| B.Tech LLB-Cyber Law | 500067119 | Ramya Mihir |



Department of Informatics

School Of Computer Science

UNIVERSITY OF PETROLEUM & ENERGY STUDIES,

DEHRADUN- 248007. Uttarakhand

Table of Contents

|  |  |  |
| --- | --- | --- |
| **Topic** | | **Page No** |
| Table of Content | | 2 |
| 1 | Introduction | 3-4 |
|  | 1.1 Purpose of the Project | 3 |
|  | 1.2 Target Beneficiary | 3 |
|  | 1.3 Project Scope | 3 |
|  | 1.4 References | 3 |
| 2 | Project Description | 5-12 |
|  | 2.1 Reference Algorithm | 5 |
|  | 2.2 Data/ Data structure | 5 |
|  | 2.3 SWOT Analysis | 6 |
|  | 2.4 Project Features | 7 |
|  | 2.5 Design and Implementation Constraints | 7 |
|  | 2.6 Design diagrams | 8 |
|  | 2.7 Assumption and Dependencies | 12 |
| 3 | System Requirements | 12-13 |
|  | 3.1 System Description | 12 |
|  | 3.2 User Interface | 13 |
|  | 3.3 Software Interface | 13 |
|  | 3.4 Database Interface | 13 |
| 4 | Non-functional Requirements | 13 |
|  | 4.1 Performance requirements | 13 |
|  | 4.2 Security requirements | 13 |
|  | 4.3 Software Quality Attributes | 14 |
| 5. | Code Snippets and Limitations | 14 |
|  | 5.1 Code Snippets | 14 |
|  | 5.2 Limitations | 23 |
| 6. | Conclusion | 23 |

1. **INTRODUCTION**
   1. **Purpose of Project:**

When COVID-19 hit the world, it altered the working pattern of all the people around the world. Along with this, there has been an exponential growth in the number of malwares and the cyber-crime rates.

Modern malwares use sophisticated techniques such as polymorphism and metamorphism to thwart the malware detection and analysis. Detecting malware on the basis of their features and behavior is critical for the computer security community. Most anti-virus depends on the signature-based detection which is relatively easy to evade and is ineffective for zero-day exploit-based malwares.

Also, it has been noticed that the Internet has become a vital part in our lives with increased usage of services like online banking, online reservation etc., and our dependence on the Internet is expected to grow. With the rise of the Internet, there has been huge growth in the amount of malwares in the world.

With this project, we provide a new approach to identify malwares using static analysis, i.e. without executing. With the help of different machine learning models, we will identify malwares if present in any file, in order to prevent any further attacks.

* 1. **Target Beneficiary:**

The target audience and the people who will majorly get benefitted from this project are the students as well as the working professionals who are these days working in online mode due to the pandemic.

This application will promote an easy use to identify the files that they receive over emails, or sms, or any other e-mode, to scan before opening any malware file and get trapped. So, it is for mainly all the students, and professionals, who are more likely to be active on internet.

* 1. **Project Scope:**

The project scope is to create an easy platform for malware detection in real-time environment where the students, working professionals, and even the new internet generation people can scan and check for malwares in any file, in order to prevent themselves from getting trapped in cyber world. This project is also to create cyber awareness among the people so that the crimes that are increasing these days due to internet, are reduced to a smaller extend.

Our main objective is to Identify and classify malwares using static analysis i.e. without executing with the help of Machine learning models. So, we are basically providing a software which will run and ask for the file to be checked for malwares and notify the user for either to open it or not.

* 1. **References:**

[1] Machine Learning Models and applications

(https://media.kaspersky.com/en/enterprise-security/Kaspersky-Lab-Whitepaper-Machine-Learning.pdf)

[2] ML models required for Malware Analysis

(https://www.ccsinet.com/blog/machine-learning-malware-analysis/)

[3] Working method for Malware Analysis

(https://sectigostore.com/blog/malware-analysis-what-it-is-how-it-works/)

[4] Malware Detection and Removal Types and Introduction

(https://www.imperva.com/learn/application-security/malware-detection-and-removal/)

[5] Images

(https://link.springer.com/chapter/10.1007/978-3-319-63673-3\_21) (https://ai.plainenglish.io/malware-detection-using-hybrid-analysis-4b0ef471e844) (https://www.researchgate.net/publication/289674086\_Detecting\_Malware\_for\_Android\_Platform\_An\_SVM-\_based\_Approach/figures?lo=1&utm\_source=google&utm\_medium=organic) (https://www.semanticscholar.org/paper/Malware-detection-using-assembly-code-and-control-Anju-Harmya/96b22134053f4a0cce2a8f9961131a42af831b8f)

1. **PROJECT DESCRIPTION**
   1. **Reference Algorithm**

The project basically follows quite a simple methodology and the reference algorithm explaining our methodology is as follows.

Steps:

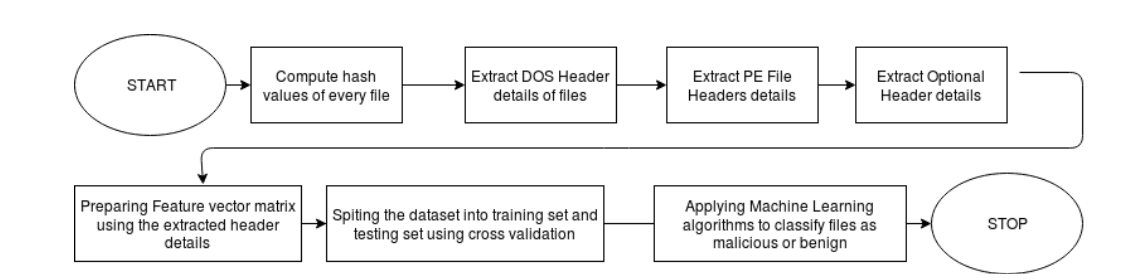
1. START
2. Compute hash values of every file and check whether any of the file in the corpus is duplicate and then remove the duplicate files.
3. Extract header details of the binaries with the help of PE File module functions of python for the analysis purpose.
4. The following header details are extracted

* DOS header
* PE File header
* Optional header

1. Prepare the feature vector matrix by selecting best features for training and testing purpose of the dataset.
2. With Cross-Validation split the dataset into training and testing set.
3. Apply ML algorithms to classify files as malwares and benign Algorithms used to evaluate:

* K-Nearest Neighbors
* Decision Trees
* Random Forest
* Logistic Regression
* SVM (Support Vector Machines)

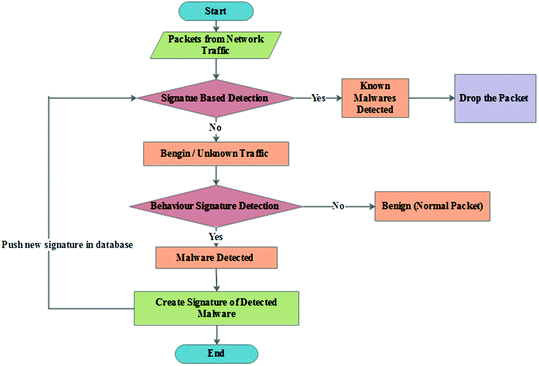
1. STOP



* 1. **Data/ Data Structure**
* **Static Analysis**: Decompiling, Parsing, Features Generations
* **Dynamic Analysis**: Emulation, Log Extraction, Feature Generation
* **Feature Selection**
* **ML and Malware Detection**: Model Building, Testing, Classification Report

This is the flowchart which shows how the file will be checked for it. First, its packets will be checked from the network traffic, then using the ML Models and algorithms, it which if it is malicious, if found malicious, it will drop that packet, else if it is not known, then will check its behavior and decide, else it will be normal to open.

This project will use libraries like Python PE file, hashlib, pandas, DOS Header in the program.



2.2.1 Flowchart

* 1. **SWOT Analysis**
* **Strengths**

The Support Vector machine and K-Nearest neighbor that are used to in the program, train the system which helps in providing higher accuracy when compared to other machine learning and deep learning models.

* **Weakness**

The model is quite heavy and requires a lot of processing power for testing stage. The available dataset for network trafﬁc are not very large, which affects the accuracy. In the project, we have not implemented the procedures for ﬁrewall protection, load description and handling, DDoS attacks etc. The rules for forwarding and dropping packets are also described, but are subject to human error.

* **Opportunities**

The work can be done in anomaly-based Network Intrusion Detection Systems (NIDS), ﬁrewall protection, load description and handling, and their respective Northbound and Southbound APIs. More research into this domain can help improve accuracy and handle various types of attacks.

* **Threats**

This project can only work with network packets. Adversarial malware attacks can bypass the discriminator classiﬁcation schema. Zero-day DDoS attacks can mostly cause network failure, as human errors could be present in ﬂow table handling.

* 1. **Project Features**

The application will allow the user to select the file from the database of the user.

Then, it will check and scan the file through different algorithms applied in it.

If the file is safe to use, it will show that the file can be opened, else will warn the user not to open the file as it contains a malicious apk file.

It requires an active internet connection of the user.

This project will help in reducing the phishing attacks, where the criminal sends malicious files through mails and the people get trapped.

This project will also help in avoiding the cross-site scripting attacks, where the criminal can send a malicious JavaScript code within the file.

It performs malware detection, where models will be trained on large corpus of executable using good set of discriminative predictors extracted through static analysis.

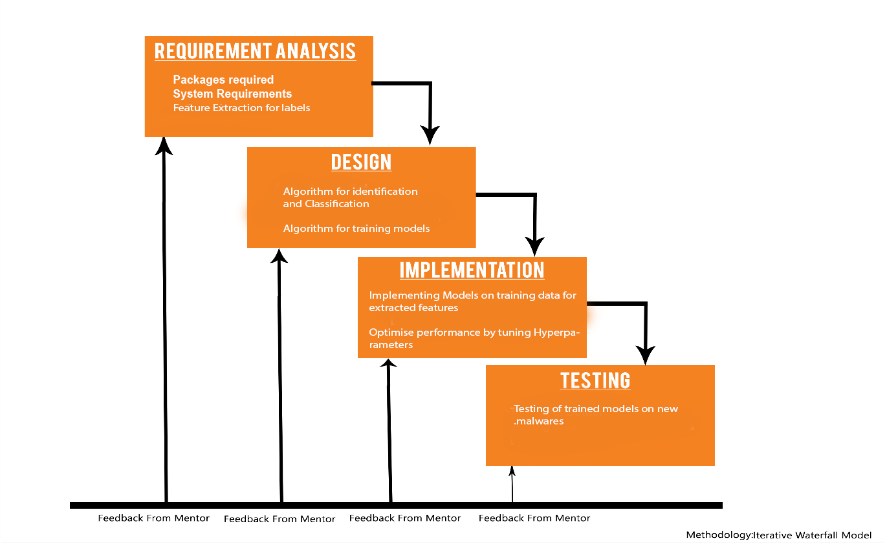
* 1. **Design and Implementation Constraints**

**Implementation Constraints:**

1. The model will only work on Windows Portable Executables (PEs). The number of available adversarial malwares is limited so it will restrict the robustness of the network.
2. The input test ﬁles will be correctly labeled, and there can be no misclassiﬁed PEs. Each input ﬁle will have a Windows API call. Processor and memory requirements are met beforehand.

**Design:**

* After the hash value for every file is computed.
* Header details are extracted.
* Applying the machine learning algorithms.
* We have classified Unseen Setup files as legitimate or malicious



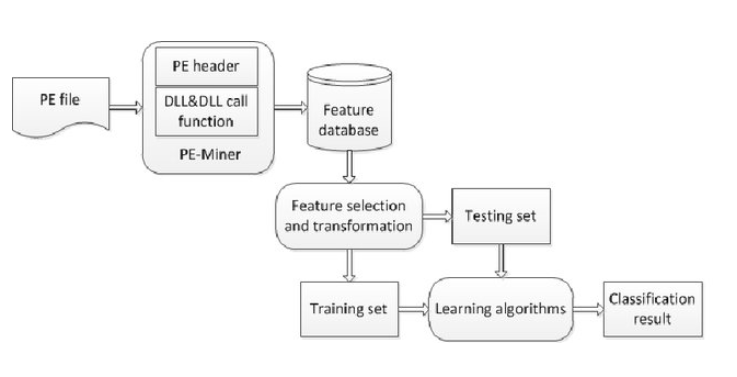
2.5.1 Waterfall Model

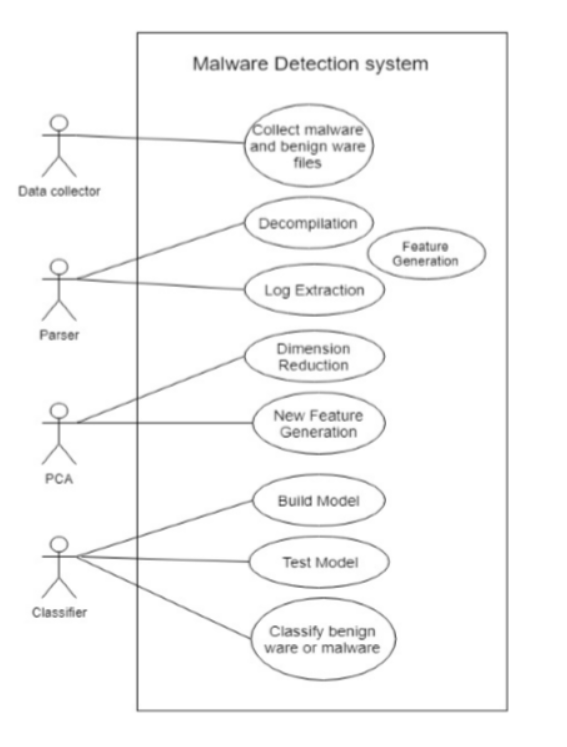
* 1. **Design Diagrams**

There are various UML Diagrams that we have created so as to explain the design and

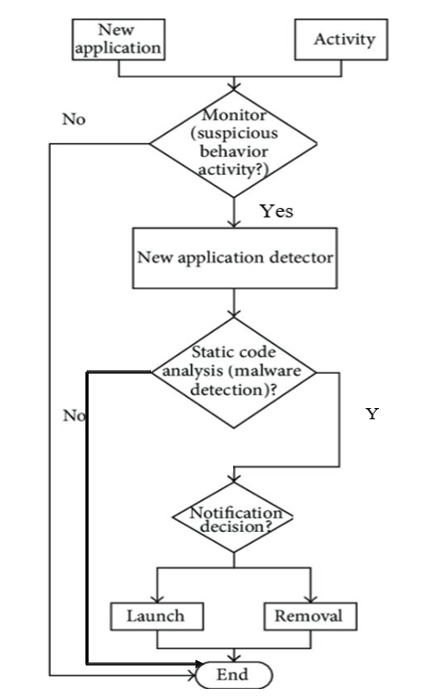
functionality of the project better.

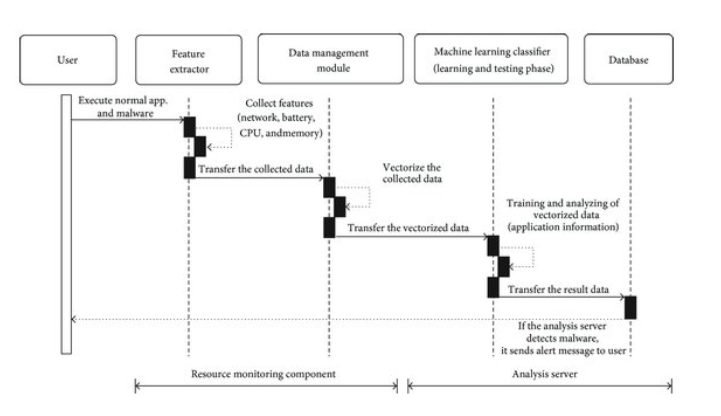
Steps:

1. Upload the file or the application to be checked for malware.
2. It will monitor the content in every file
3. If found suspicious, the detector will come into place and using the static code analysis method, will detect the malicious activity
4. Then will decide whether to give notification or not
5. Then will either launch it and tell the user, else will try to remove it if possible, and already present in the library.
6. If not found suspicious, the file is opened, and the process ends
7. END
   * 1. **Data Flow**
     2. **Use Case Diagram**



* + 1. **Activity Diagram**



* + 1. **Sequence Diagram**
  1. **Assumptions and Dependencies**

1. The basic assumptions includes that the user should know how to select a file before opening it.
2. The user should understand the output of whether the file is malicious or not; if malicious, what is that file name.
3. The project is dependent on the internet connectivity of the user. The project requires internet connection for multiple users.
4. **SYSTEM REQUIREMENTS**
   1. **System Description**

Malware includes virus, ransomware, rootkits, trojans and a malware attack can even adversely affect a business and its operations. Appropriate security measures must be put in place by businesses to malware analysis tools as an incident response plan that will provide a proper procedure to ensure there are the recovery time and reduced costs.

Malware analysis plays a vital role in aiding the security team to understand the extent of the incident along with identifying the hosts and systems that have been affected. With the help of the report generated from malware analysis, an organization can mitigate any vulnerabilities and prevent any additional compromises.

Thus, through this platform, we provide the organizations and companies an option of mitigating any future malware attacks planned by the attackers to stay protected. This project is done on a google collaborator, where the file received by the corporates can be checked for any vulnerability and thus, prevent it by either not opening it, or by bypassing that vulnerability. Hence, the company can know whether it is completely legitimate or partially legitimate or is not at all safe to access that file, thereby aiding security, and integrity to the user.

* 1. **User Interface**

Simple and easy to use system should be there, where the user is able to detect malware using the APK dataset. Machine learning is used to build the model using the selected features as input. Comparative analysis is done and a classiﬁcation report is generated.

* 1. **Software Interface**
* Programming Language: Python
* Operating System: Windows/Linux/Macintosh
* Dataset: APK dataset
  1. **Database Interface**
* Google Collaborator

1. **NON-FUNCTIONAL REQUIREMENTS**
   1. **Performance Requirements**

* The system should be based on web and has to run form a web server
* The system should take initial load time depending on internet connection strength of user which also depends on the media size uploaded on the application.
  1. **Security Requirements**
* The system must check for authorization.
* The system should check for malware file.
* The system must automatically warn the user if found malicious.
  1. **Software Quality Attributes**
* AVAILABILITY: Malware Detection will be for each user every time.
* CORRECTNESS: The detector should always connect to the correct malware apk file.
* MAINTAINABILITY: Detector should maintain its state even if any older or newer files is checked.
* USABILITY: Detector should satisfy a maximum number of users needs of having a malware free file.

1. **CODE SNIPPETS and LIMITATIONS**
   1. **Code Snippets**

**Importing Libraries**

# from google.colab import drive

# drive.mount('/content/drive')

# !cp "/content/drive/My Drive/data.csv" "data.csv"

# from google.colab import drive

# drive.mount('/content/drive')

# !cp "/content/drive/My Drive/classifier.pkl" "classifier.pkl"

# from google.colab import drive

# drive.mount('/content/drive')

# !cp "/content/drive/My Drive/features.pkl" "features.pkl"

# from google.colab import drive

# drive.mount('/content/drive')

# !cp "/content/drive/My Drive/7z1604-x64.exe" "7z1604-x64.exe"

# from google.colab import drive

# drive.mount('/co        res['BaseOfData'] = pe.OPTIONAL\_ntent/drive')

# !cp "/content/drive/My Drive/vlc.exe" "vlc.exe"

from google.colab import drive

drive.mount('/content/drive')

csvpath = "/content/drive/MyDrive/major 1/data.csv"

!pip install pefile

import os

import pandas

import numpy

import pickle

import pefile

import sklearn.ensemble as ek

import array

import math

import pickle

from sklearn.externals import joblib

import sys

import argparse

from sklearn.model\_selection import cross\_validate

from sklearn import tree, linear\_model

from sklearn.model\_selection import KFold, cross\_val\_score

from sklearn.feature\_selection import SelectFromModel

from sklearn.externals import joblib

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import confusion\_matrix

from sklearn.pipeline import make\_pipeline

from sklearn import preprocessing

from sklearn import svm

from sklearn.linear\_model import LinearRegression

Graphical user interface, text, application, email

Description automatically generated

dataset = pandas.read\_csv(csvpath, sep='|', low\_memory=False)

dataset.head()

Graphical user interface, application

Description automatically generated

dataset.describe()

Graphical user interface, table

Description automatically generated

dataset.groupby(dataset['legitimate']).size()Graphical user interface, text, application

Description automatically generated

X = dataset.drop(['Name','md5','legitimate'],axis=1).values

y = dataset['legitimate'].values

**TEST CASE 1- Checking whther the file is uploaded or not**

extratrees = ek.ExtraTreesClassifier().fit(X,y)

model = SelectFromModel(extratrees, prefit=True)

X\_new = model.transform(X)

nbfeatures = X\_new.shape[1]

nbfeatures

Graphical user interface, application

Description automatically generated with medium confidence

from sklearn import model\_selection

from sklearn.model\_selection import cross\_validate

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_new, y ,test\_size=0.2)

features = []

index = numpy.argsort(extratrees.feature\_importances\_)[::-1][:nbfeatures]

for f in range(nbfeatures):

    print("%d. feature %s (%f)" % (f + 1, dataset.columns[2+index[f]], extratrees.feature\_importances\_[index[f]]))

   features.append(dataset.columns[2+f])

Graphical user interface, text, application, email

Description automatically generated

from matplotlib.pyplot import \*

import matplotlib.pyplot as plt

model = { "DecisionTree":tree.DecisionTreeClassifier(max\_depth=10),      "GradientBoosting":ek.GradientBoostingClassifier(n\_estimators=50),

         "LinearRegression":LinearRegression(),

         "RandomForest":ek.RandomForestClassifier(n\_estimators=50),

         "Gaussian Naive Bayes":GaussianNB()

}

results = {}

for algo in model:

    clf = model[algo]

    clf.fit(X\_train,y\_train)

    score = clf.score(X\_test,y\_test)

    print ("%s : %s " %(algo, score))

    results[algo] = score

    z = 0

  plt.plot(z, score, color='green', linestyle='dashed', linewidth = 3,

  marker='o', markerfacecolor='blue', markersize=8)

    plt.ylim(0,2)

    plt.xlim(-1,1)

    plt.show()

Graphical user interface, application

Description automatically generated

%cd /content/drive/MyDrive/major 1/

winner = max(results, key=results.get)

joblib.dump(model[winner],'classifier.pkl')

open('features.pkl', 'wb').write(pickle.dumps(features))

clf = model[winner]

res = clf.predict(X\_new)

mt = confusion\_matrix(y, res)

print("False positive rate : %f %%" % ((mt[0][1] / float(sum(mt[0])))\*100))

print('False negative rate : %f %%' % ( (mt[1][0] / float(sum(mt[1]))\*100)))

clf = joblib.load('classifier.pkl')

features = pickle.loads(open(os.path.join('features.pkl'),'rb').read())

import pefile

import os

import array

import math

import pickle

from sklearn.externals import joblib

import sys

import argparse

file\_path = "/content/drive/MyDrive/major 1/7z1604-x64.exe"

**TEST CASE 2- Checking whether the file has the data/ content in it or not**

def get\_entropy(data):

    if len(data) == 0:

        return 0.0

occurences = array.array('L', [0] \* 256)

    for x in data:

occurences[x if isinstance(x, int) else ord(x)] += 1

    entropy = 0

    for x in occurences:

        if x:

            p\_x = float(x) / len(data)

            entropy -= p\_x \* math.log(p\_x, 2)

    return entropy

def get\_resources(pe):

    """Extract resources :

    [entropy, size]"""

    resources = []

    if hasattr(pe, 'DIRECTORY\_ENTRY\_RESOURCE'):

        try:

            for resource\_type in pe.DIRECTORY\_ENTRY\_RESOURCE.entries:

                if hasattr(resource\_type, 'directory'):

                    for resource\_id in resource\_type.directory.entries:

                        if hasattr(resource\_id, 'directory'):

                            for resource\_lang in resource\_id.directory.entries:

                                data = pe.get\_data(resource\_lang.data.struct.OffsetToData                      resource\_lang.data.struct.Size)

                                size = resource\_lang.data.struct.Size

                                entropy = get\_entropy(data)

                                resources.append([entropy, size])

        except Exception as e:

            return resources

    return resources

def get\_version\_info(pe):

    """Return version infos"""

    res = {}

    for fileinfo in pe.FileInfo:

        if fileinfo.Key == 'StringFileInfo':

            for st in fileinfo.StringTable:

                for entry in st.entries.items():

                    res[entry[0]] = entry[1]

        if fileinfo.Key == 'VarFileInfo':

            for var in fileinfo.Var:

                res[var.entry.items()[0][0]] = var.entry.items()[0][1]

    if hasattr(pe, 'VS\_FIXEDFILEINFO'):

        res['flags'] = pe.VS\_FIXEDFILEINFO.FileFlags

        res['os'] = pe.VS\_FIXEDFILEINFO.FileOS

        res['type'] = pe.VS\_FIXEDFILEINFO.FileType

        res['file\_version'] = pe.VS\_FIXEDFILEINFO.FileVersionLS

        res['product\_version'] = pe.VS\_FIXEDFILEINFO.ProductVersionLS

        res['signature'] = pe.VS\_FIXEDFILEINFO.Signature

        res['struct\_version'] = pe.VS\_FIXEDFILEINFO.StrucVersion

    return res

**TEST CASE 3- Checking whether the file is malicious or legitimate**

# extract the info for a given file

def extract\_infos(fpath):

    res = {}

    pe = pefile.PE(file\_path)

    res['Machine'] = pe.FILE\_HEADER.Machine

    res['SizeOfOptionalHeader'] = pe.FILE\_HEADER.SizeOfOptionalHeader

    res['Characteristics'] = pe.FILE\_HEADER.Characteristics

    res['MajorLinkerVersion'] = pe.OPTIONAL\_HEADER.MajorLinkerVersion

    res['MinorLinkerVersion'] = pe.OPTIONAL\_HEADER.MinorLinkerVersion

    res['SizeOfCode'] = pe.OPTIONAL\_HEADER.SizeOfCode

    res['SizeOfInitializedData'] = pe.OPTIONAL\_HEADER.SizeOfInitializedData

    res['SizeOfUninitializedData'] = pe.OPTIONAL\_HEADER.SizeOfUninitializedData

    res['AddressOfEntryPoint'] = pe.OPTIONAL\_HEADER.AddressOfEntryPoint

    res['BaseOfCode'] = pe.OPTIONAL\_HEADER.BaseOfCode

    try:

HEADER.BaseOfData

    except AttributeError:

        res['BaseOfData'] = 0

    res['ImageBase'] = pe.OPTIONAL\_HEADER.ImageBase

    res['SectionAlignment'] = pe.OPTIONAL\_HEADER.SectionAlignment

    res['FileAlignment'] = pe.OPTIONAL\_HEADER.FileAlignment

    res['MajorOperatingSystemVersion'] = pe.OPTIONAL\_HEADER.MajorOperatingSystemVersion

    res['MinorOperatingSystemVersion'] = pe.OPTIONAL\_HEADER.MinorOperatingSystemVersion

    res['MajorImageVersion'] = pe.OPTIONAL\_HEADER.MajorImageVersion

    res['MinorImageVersion'] = pe.OPTIONAL\_HEADER.MinorImageVersion

    res['MajorSubsystemVersion'] = pe.OPTIONAL\_HEADER.MajorSubsystemVersion

    res['MinorSubsystemVersion'] = pe.OPTIONAL\_HEADER.MinorSubsystemVersion

    res['SizeOfImage'] = pe.OPTIONAL\_HEADER.SizeOfImage

    res['SizeOfHeaders'] = pe.OPTIONAL\_HEADER.SizeOfHeaders

    res['CheckSum'] = pe.OPTIONAL\_HEADER.CheckSum

    res['Subsystem'] = pe.OPTIONAL\_HEADER.Subsystem

    res['DllCharacteristics'] = pe.OPTIONAL\_HEADER.DllCharacteristics

    res['SizeOfStackReserve'] = pe.OPTIONAL\_HEADER.SizeOfStackReserve

    res['SizeOfStackCommit'] = pe.OPTIONAL\_HEADER.SizeOfStackCommit

    res['SizeOfHeapReserve'] = pe.OPTIONAL\_HEADER.SizeOfHeapReserve

    res['SizeOfHeapCommit'] = pe.OPTIONAL\_HEADER.SizeOfHeapCommit

    res['LoaderFlags'] = pe.OPTIONAL\_HEADER.LoaderFlags

    res['NumberOfRvaAndSizes'] = pe.OPTIONAL\_HEADER.NumberOfRvaAndSizes

    # Sections

    res['SectionsNb'] = len(pe.sections)

    entropy = list(map(lambda x: x.get\_entropy(), pe.sections))

    res['SectionsMeanEntropy'] = sum(entropy) / float(len(entropy))

    res['SectionsMinEntropy'] = min(entropy)

    res['SectionsMaxEntropy'] = max(entropy)

    raw\_sizes = list(map(lambda x: x.SizeOfRawData, pe.sections))

    res['SectionsMeanRawsize'] = sum(raw\_sizes) / float(len(raw\_sizes))

    res['SectionsMinRawsize'] = min(raw\_sizes)

    res['SectionsMaxRawsize'] = max(raw\_sizes)

    virtual\_sizes = list(map(lambda x: x.Misc\_VirtualSize, pe.sections))

    res['SectionsMeanVirtualsize'] = sum(virtual\_sizes) / float(len(virtual\_sizes))

    res['SectionsMinVirtualsize'] = min(virtual\_sizes)

    res['SectionMaxVirtualsize'] = max(virtual\_sizes)

    # Imports

    try:

        res['ImportsNbDLL'] = len(pe.DIRECTORY\_ENTRY\_IMPORT)

     imports = sum([x.imports for x in pe.DIRECTORY\_ENTRY\_IMPORT], [])

        res['ImportsNb'] = len(imports)

        res['ImportsNbOrdinal'] = len(list(filter(lambda x: x.name is None, imports)))

    except AttributeError:

        res['ImportsNbDLL'] = 0

        res['ImportsNb'] = 0

        res['ImportsNbOrdinal'] = 0

    # Exports

    try:

        res['ExportNb'] = len(pe.DIRECTORY\_ENTRY\_EXPORT.symbols)

    except AttributeError:

        # No export

        res['ExportNb'] = 0

    # Resources

    resources = get\_resources(pe)

    res['ResourcesNb'] = len(resources)

    if len(resources) > 0:

        entropy = list(map(lambda x: x[0], resources))

        res['ResourcesMeanEntropy'] = sum(entropy) / float(len(entropy))

        res['ResourcesMinEntropy'] = min(entropy)

        res['ResourcesMaxEntropy'] = max(entropy)

        sizes = list(map(lambda x: x[1], resources))

        res['ResourcesMeanSize'] = sum(sizes) / float(len(sizes))

        res['ResourcesMinSize'] = min(sizes)

        res['ResourcesMaxSize'] = max(sizes)

    else:

        res['ResourcesNb'] = 0

        res['ResourcesMeanEntropy'] = 0

        res['ResourcesMinEntropy'] = 0

        res['ResourcesMaxEntropy'] = 0

        res['ResourcesMeanSize'] = 0

        res['ResourcesMinSize'] = 0

        res['ResourcesMaxSize'] = 0

    # Load configuration size

    try:

        res['LoadConfigurationSize'] = pe.DIRECTORY\_ENTRY\_LOAD\_CONFIG.struct.Size

    except AttributeError:

        res['LoadConfigurationSize'] = 0

    # Version configuration size

    try:

        version\_infos = get\_version\_info(pe)

        res['VersionInformationSize'] = len(version\_infos.keys())

    except AttributeError:

        res['VersionInformationSize'] = 0

    return res

if \_\_name\_\_ == '\_\_main\_\_':

    clf = joblib.load('classifier.pkl')

    features = pickle.loads(open(os.path.join('features.pkl'), 'rb').read())

    data = extract\_infos(sys.argv[1])

    pe\_features = list(map(lambda x: data[x], features))

    res = clf.predict([pe\_features])[0]

    print('The file is %s' % (['malicious', 'legitimate'][res]))

**Output**

Text, application

Description automatically generated

**5.2 Limitations**

1. File must be accessed from google drive,
2. Currently, this program is only able to detect malicious content in the tabular form of data, hence no files other than CSV or XLSX are possible to detect.
3. It requires authentication every time it compiles and runs, although it is an additional security feature but might be a limitation for access at different devices.
4. The program is only executable on google collab due to dependency on google drive.
   1. **CONCLUSION**

In this world of digitalization, where advancements have made people so comfortable with the digital services, that they don’t need to travel out and stand in long queues to get a small work done. But with these advancements, people are also prone to several offences and crimes that might have been planned against them, which occur online, and would lead to asking for ransom or doing some illegal act on dark webs, etc. These offences occur due to the slight mistake done on the part of the people due to the lack of knowledge of what files to open and what not.

This platform created by us, will help the people to know whether the document that they have received is malicious and should not be opened or are legitimate and are safe to be opened. Thus, this platform will help the people to stay cyber safe and thereby reducing the cybercrime rate.