Feasibility Report

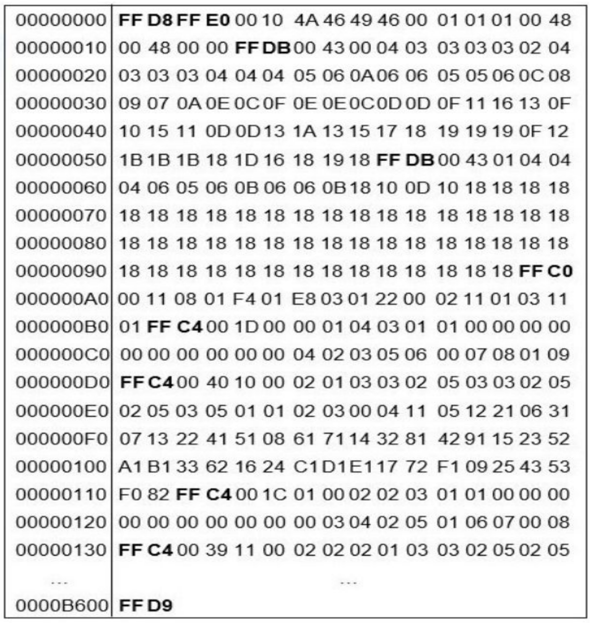
SE 303 Mini Project - Team 12

Problem Statement:

To implement a machine learning based solution for the detection of malicious JPEG images in Python.

Motivation:

Cyber attackers use files sent over the Internet to launch attacks on individuals and organizations.Files such as .exe are considered to be harmful and hence files that are considered harmless by most users such as .pdf, .docx are being increasingly used to hide malicious payloads in them. JPEG is a popular image format. Embedding malicious payloads can be done by exploiting vulnerabilities in JPEG file structure or image viewers.
The JPEG file structure contains elements called markers and these are used by attackers to hide the payloads. So by examining the relative content of these markers in malicious and benign images one can classify the image.This is the basic idea behind MalJPEG.



```
00000000 | FF D8 FF E0 00 10 4A 46 49 46 00 01 01 01 00 48
00000010 | 00 48 00 00 FF DB 00 43 00 04 03 03 03 03 02 04
00000020 | 03 03 03 04 04 04 05 06 0A 06 06 05 05 06 0C 08
00000030 | 09 07 0A 0E 0C 0F 0E 0E 0C 0D 0D 0F 11 16 13 0F
00000040 | 10 15 11 0D 0D 13 1A 13 15 17 18 19 19 19 0F 12
00000050 | 1B 1B 1B 18 1D 16 18 19 18 FF DB 00 43 01 04 04
00000060 | 04 06 05 06 0B 06 06 0B 18 10 0D 10 18 18 18 18
00000070 | 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18
00000080 | 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18
00000090 | 18 18 18 18 18 18 18 18 18 18 18 18 18 18 FF C0
000000A0 | 00 11 08 01 F4 01 E8 03 01 22 00 02 11 01 03 11
000000B0 | 01 FF C4 00 1D 00 00 01 04 03 01 01 00 00 00 00
000000C0 | 00 00 00 00 00 00 04 02 03 05 06 00 07 08 01 09
000000D0 | FF C4 00 40 10 00 02 01 03 03 02 05 03 03 02 05
000000E0 | 02 05 03 05 01 01 02 03 00 04 11 05 12 21 06 31
000000F0 | 07 13 22 41 51 08 61 71 14 32 81 42 91 15 23 52
00000100 | A1 B1 33 62 16 24 C1 D1 E1 17 72 F1 09 25 43 53
00000110 | F0 82 FF C4 00 1C 01 00 02 02 03 01 01 00 00 00
00000120 | 00 00 00 00 00 00 00 03 04 02 05 01 06 07 00 08
00000130 | FF C4 00 39 11 00 02 02 02 01 03 03 02 05 02 05
   ...   |                      ...
0000B600 | FF D9
```

Fig1: JPEG file structure with markers

Implementation:

The steps involved in the MalJPEG solution are as follows.
1.MalJPEG receives an image as an input.
2.MalJPEG examines the image statically without invoking an image viewer.(Might have a vulnerability)
3.Traverse the JPEG file structure to extract the features.
4.Transfer the features to a machine learning based model which outputs a classification.

The MalJPEG features are based on the presence and size of specific markers within the JPEG image file structure.This is because malicious images always have something to do with markers.For example, it has been found that some malicious images contain data after the end-of-file(EOI) marker.

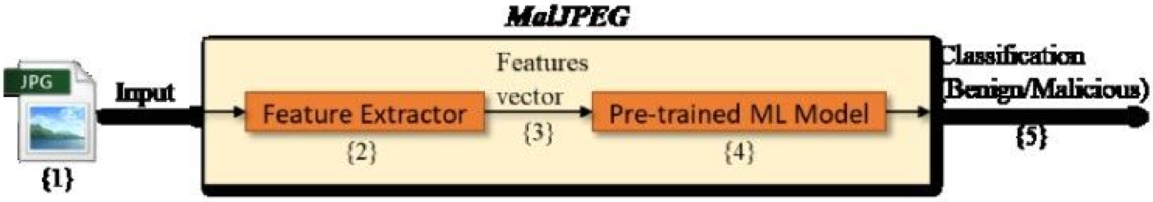
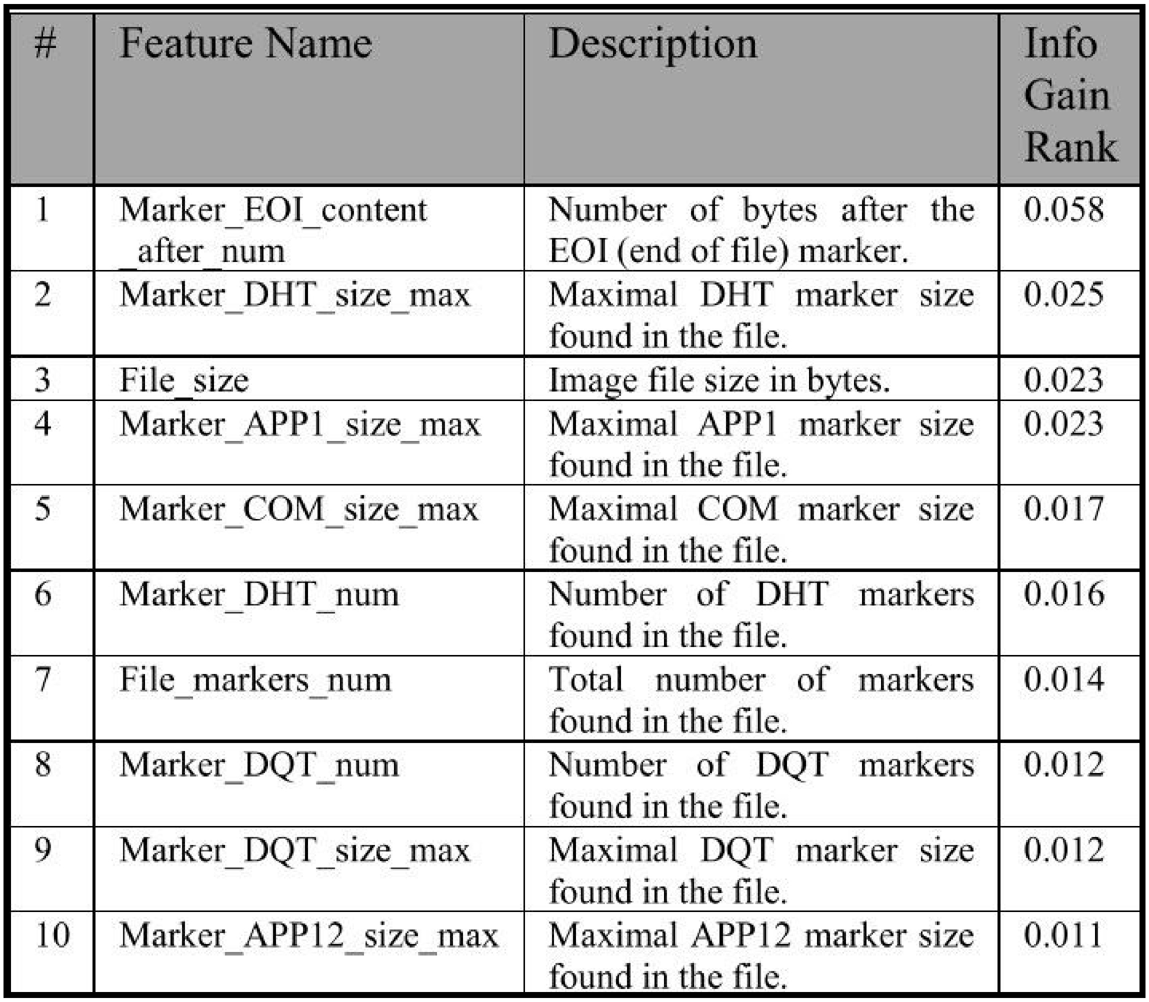


Fig 2: The MalJPEG solution



#	Feature Name	Description	Info Gain Rank
1	Marker_EOI_content _after_num	Number of bytes after the EOI (end of file) marker.	0.058
2	Marker_DHT_size_max	Maximal DHT marker size found in the file.	0.025
3	File_size	Image file size in bytes.	0.023
4	Marker_APP1_size_max	Maximal APP1 marker size found in the file.	0.023
5	Marker_COM_size_max	Maximal COM marker size found in the file.	0.017
6	Marker_DHT_num	Number of DHT markers found in the file.	0.016
7	File_markers_num	Total number of markers found in the file.	0.014
8	Marker_DQT_num	Number of DQT markers found in the file.	0.012
9	Marker_DQT_size_max	Maximal DQT marker size found in the file.	0.012
10	Marker_APP12_size_max	Maximal APP12 marker size found in the file.	0.011

Fig: MalJPEG features.

A total of 10 features are detected . These features have been decided by the researchers of the base paper after closely examining the file structure of malicious and benign images.

How will this help?

Antivirus engines can detect malware only based on its signature database. When a new malware is identified or discovered there is a time lag in updating the signature database. A machine learning solution has no such limitation and can be greatly useful.

Based on the above points, we have decided to implement this solution to the malicious JPEG problem in Python.

Team Members

Akshith Nettar Mahalinga	181IT104
Amith Bhat	181IT103
Laharish S	181IT125