MalJPEG: Machine Learning Based Solution for the detection of Malicious JPEG images.

Group 12

Akshith Nettar Mahalinga	181IT104
Amith Bhat Nekkare	181IT105
Laharish S	181IT104

Abstract

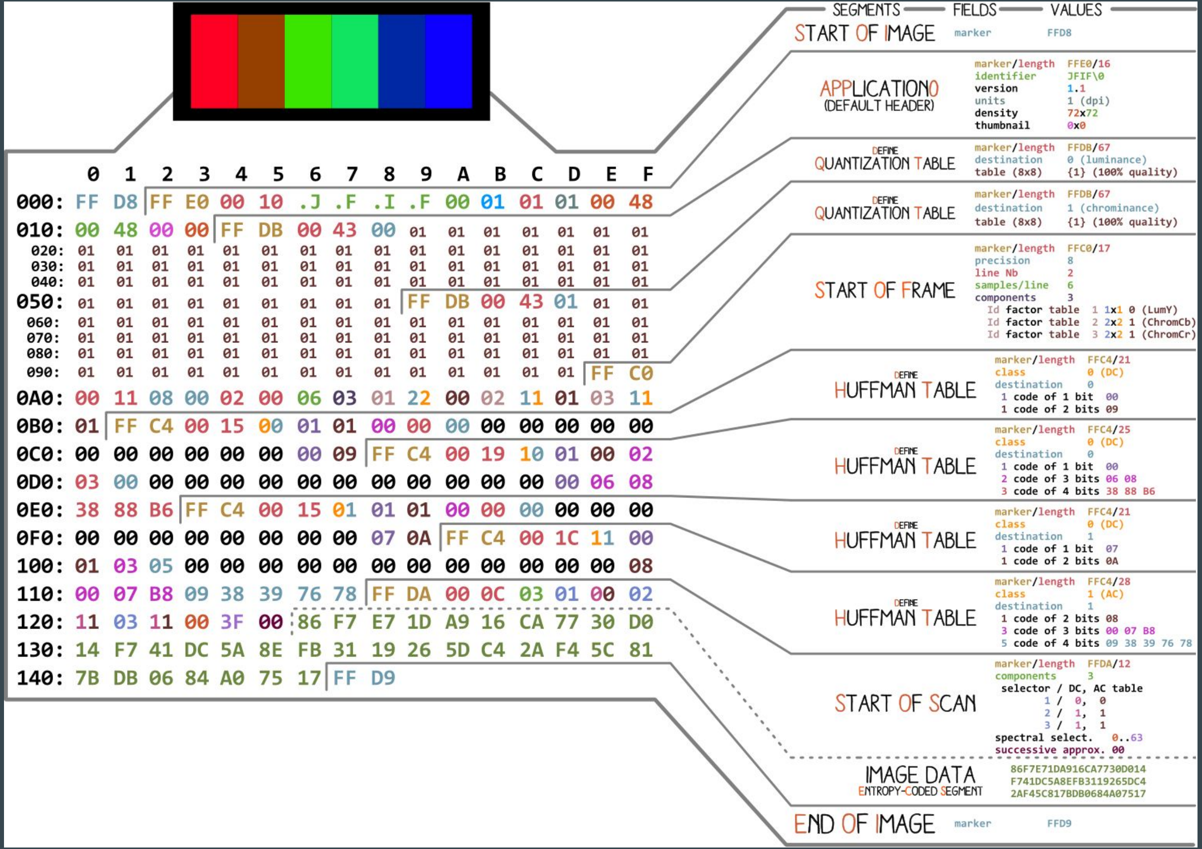
- In recent years, cyber-attacks against individuals, businesses, Images are used on a daily basis by millions of people around the world, and most users consider images to be safe for use; however, some types of images can contain a malicious payload and perform harmful actions. JPEG is the most popular image format, primarily due to its lossy compression. Because of their harmless reputation and massive use, JPEG images are used by cyber criminals as an attack vector.
- MalJPEG is the first machine learning- based solution tailored specifically at the efficient detection of unknown malicious JPEG images. MalJPEG statically extracts 10 simple yet discriminative features from the JPEG file structure and leverages them with a machine learning classifier, in order to discriminate between benign and malicious JPEG images.

Introduction

The contributions of the research work done in the base paper are as follows:

1. MalJPEG: A machine learning based solution for efficient detection of know and malicious JPEG images.
2. MalJPEG features: a compact set of 10 features for efficient detection.
3. The creation of large labeled collection of benign and malicious JPEG images.

JPEG file structure



```
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
000: FF D8 FF E0 00 10 .J .F .I .F 00 01 01 01 00 48
010: 00 48 00 00 FF DB 00 43 00 01 01 01 01 01 01 01
020: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
030: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
040: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
050: 01 01 01 01 01 01 01 01 01 FF DB 00 43 01 01 01
060: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
070: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
080: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
090: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 FF C0
0A0: 00 11 08 00 02 00 06 03 01 22 00 02 11 01 03 11
0B0: 01 FF C4 00 15 00 01 01 00 00 00 00 00 00 00 00
0C0: 00 00 00 00 00 00 00 09 FF C4 00 19 10 01 00 02
0D0: 03 00 00 00 00 00 00 00 00 00 00 00 00 00 06 08
0E0: 38 88 B6 FF C4 00 15 01 01 01 00 00 00 00 00 00
0F0: 00 00 00 00 00 00 00 00 07 0A FF C4 00 1C 11 00
100: 01 03 05 00 00 00 00 00 00 00 00 00 00 00 00 08
110: 00 07 B8 09 38 39 76 78 FF DA 00 0C 03 01 00 02
120: 11 03 11 00 3F 00 86 F7 E7 1D A9 16 CA 77 30 D0
130: 14 F7 41 DC 5A 8E FB 31 19 26 5D C4 2A F4 5C 81
140: 7B DB 06 84 A0 75 17 FF D9
```

SEGMENTS — FIELDS — VALUES

START OF IMAGE

```
marker          FFD8
```

APPLICATION0 (DEFAULT HEADER)

```
marker/length   FFE0/16
identifier      JFIF\0
version         1.1
units           1 (dpi)
density         72x72
thumbnail       0x0
```

DEFINE QUANTIZATION TABLE

```
marker/length   FFDB/67
destination     0 (luminance)
table (8x8)     {1} (100% quality)
```

DEFINE QUANTIZATION TABLE

```
marker/length   FFDB/67
destination     1 (chrominance)
table (8x8)     {1} (100% quality)
```

START OF FRAME

```
marker/length   FFC0/17
precision       8
line Nb         2
samples/line    6
components      3
  Id factor table  1 1x1 0 (LumY)
  Id factor table  2 2x2 1 (ChromCb)
  Id factor table  3 2x2 1 (ChromCr)
```

DEFINE HUFFMAN TABLE

```
marker/length   FFC4/21
class           0 (DC)
destination     0
 1 code of 1 bit  00
 1 code of 2 bits 09
```

DEFINE HUFFMAN TABLE

```
marker/length   FFC4/25
class           0 (DC)
destination     0
 1 code of 1 bit  00
 2 code of 3 bits 06 08
 3 code of 4 bits 38 88 B6
```

DEFINE HUFFMAN TABLE

```
marker/length   FFC4/21
class           0 (DC)
destination     1
 1 code of 1 bit  07
 1 code of 2 bits 0A
```

DEFINE HUFFMAN TABLE

```
marker/length   FFC4/28
class           1 (AC)
destination     1
 1 code of 2 bits 08
 3 code of 3 bits 00 07 B8
 5 code of 4 bits 09 38 39 76 78
```

START OF SCAN

```
marker/length   FFDA/12
components      3
 selector / DC, AC table
        1 /  0,  0
        2 /  1,  1
        3 /  1,  1
spectral select.   0..63
successive approx. 00
```

IMAGE DATA
ENTROPY-CODED SEGMENT

```
86F7E71DA916CA7730D014
F741DC5A8EFB3119265DC4
2AF45C817BDB0684A07517
```

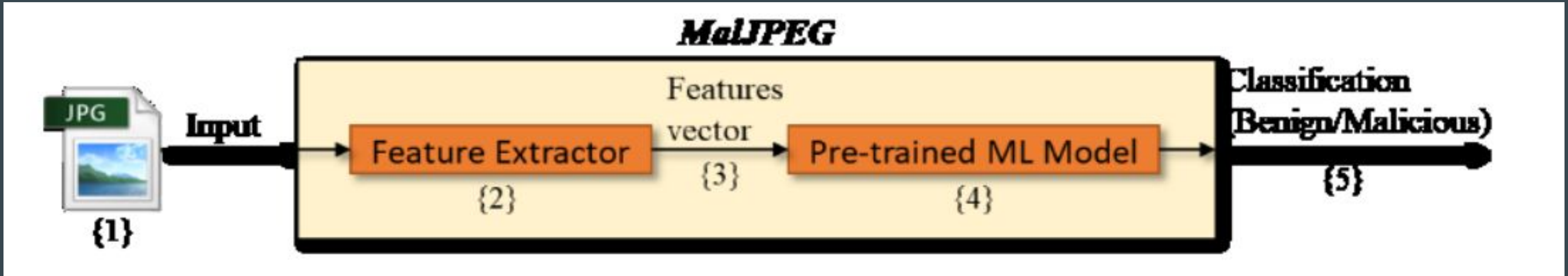
END OF IMAGE

```
marker          FFD9
```

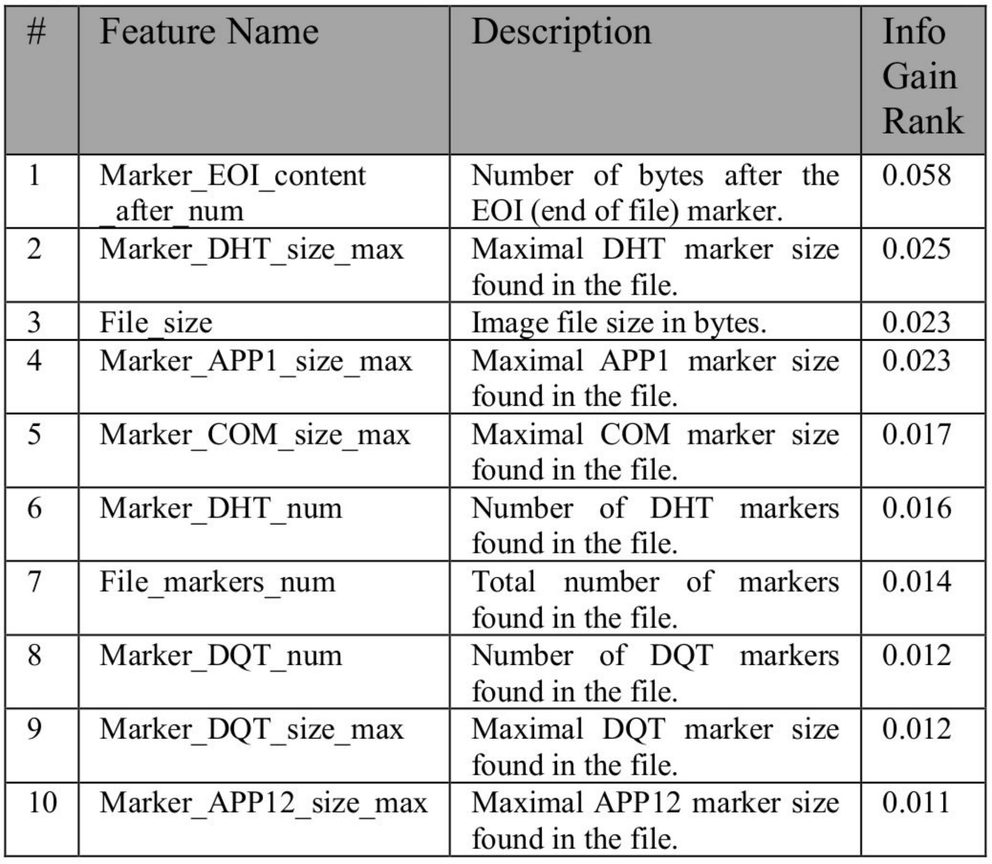
Embedding malicious payload

- Vulnerability Exploitation
- Steganography

Methodology



MalJPEG features



#	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

Work Done

- Sourcing of Malicious and Benign Images
- MalJPEG Feature Extractor :
    - Scans through the JPEG image and detects the various markers.
    - Creates a CSV file containing all 10 features proposed and another column containing an indicator which tells if the image is benign or malicious.
    - This column will be used as our prediction target.
- Generic Feature Extractor - MinHash :
    - Scans through all the images to be checked and writes their bytecode into a file
    - For each image, create a set of “shingles”(a group of words), and hash them to a CRC32 hash(for easier computation)
    - Calculate the MinHash signature for each image by hashing the shingles in the image and selecting the lowest hash value.
    - The function returns the set of signatures for each image as output.

Conclusion

For the mid-semester evaluation of the project, we have sourced the malicious images and also implemented two methods to extract features - a generic feature extractor called MinHash , and the feature extractor proposed by the paper’s authors.

By the end-semester evaluation, we will implement two more feature extraction methods(based on histograms), train selected ML models with the extracted features, and compare them using a variety of criteria.