MalJPEG: Machine Learning Based Solution For Detection Of Malicious Images

Software Engineering (IT303)

Submitted by

Akshith N M (181IT104)
Amith Bhat Nekkare (181IT105)
Laharish S (181IT125)

V SEM B.Tech (IT)

Under the guidance of

Dr.Biju R Mohan

Manjunath K Vanahalli
Dept of IT,NITK Surathkal

in partial fulfillment for the award of the degree of

Bachelor of Technology
in
Information Technology
at

Department of Information Technology

National Institute of Technology Karnataka, Surathkal.

October 2020

ABSTRACT

In recent years, cyber-attacks against individuals, businesses, and organizations have increased. Cyber criminals are always looking for effective vectors to deliver malware to victims in order to launch an attack. 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. It is used by almost everyone, from individuals to large organizations, and can be found on almost every device (on digital cameras and smartphones, websites, social media, etc.). Because of their harmless reputation, massive use, and high potential for misuse, JPEG images are used by cyber criminals as an attack vector. Thus, we have implemented MalJPEG, 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.

TABLE OF CONTENTS

INTRODUCTION

Cyber attacks targeting individuals, businesses, and organizations have increased in recent years. Infosecurity magazine declared that cyber attacks doubled in 2017. Attackers are constantly searching for new and effective ways to launch attacks and deliver a malicious payload to victims. Files sent via the Internet have often served as a means of accomplishing this. Since executable files (i.e., *.exe) are known to be dangerous, attackers are increasingly using non-executable files (e.g., *.pdf, *.docx, etc.) which are mistakenly considered to be safe for use by most users. Some non-executable files allow an attacker to run arbitrary malicious code on the targeted victim machine when the file is opened. JPEG (Joint Photographic Experts Group) is the most popular image format, mainly because of its lossy compression. JPEG images are used by almost everyone, from individuals to large enterprises, and on various platforms. JPEG images can be found on computers (personal images, documents), devices (smartphones, digital cameras, etc.), and in cyberspace. Due to their harmless reputation, massive use, and high potential for misuse, cyber criminals use JPEG images as an attack vector in order to deliver their malicious payload to the victim device. The ability to detect malicious JPEG images has great importance as JPEG images are widely used by individuals and businesses. Existing endpoint defense solutions which are based on signatures (e.g., antivirus), can only detect known malware based on their signature database. When a new malware, or new variant of existing malware appears, there is a time lag until these defense solutions update their clients with the new signature—a time in which the clients are vulnerable to the new malware. In contrast, in recent years, machine learning (ML) algorithms have demonstrated their ability to detect both known and unknown malware in various domains, particularly for the detection of malware in various types of files . However, to the best of our knowledge, machine learning methods have not been employed for the detection of malicious JPEG images. In this paper, we present MalJPEG, a machine learning based solution for efficient detection of unknown malicious JPEG images. MalJPEG extracts 10 simple but 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. We evaluate MalJPEG extensively on a real-world representative

collection of benign and malicious JPEG images. We also compare MalJPEG features to features extracted using several existing generic feature extraction methods.

JPEG stands for Joint Photographic Experts Group, which has become the most popular image format on the Web. In 1992, JPEG became an international standard for compressing digital still images. JPEG files usually have a filename extension of *.jpg or *.jpeg. A JPEG image file is a binary file which consists of a sequence of segments. Segments can be contained in other segments hierarchically. Each segment begins with a two-byte indicator called a ‘‘marker.’’ The markers help divide the file into different segments. A marker’s first byte is 0xFF (hexadecimal representation; the second byte may have any value except 0x00 and 0xFF. The marker indicates the type of data stored in the segment. Segment types are assigned names based on their definition or purpose; for example, the name of 0xFFD9 is OI, and the name of 0xFFFE is COM. Segment types 0xFF01 and 0xFF0-0xFF9 consist entirely of the two-byte marker; all other markers are followed by a two-byte integer indicating the size of the segment, followed by the payload data contained in the segment. Table 1 presents the possible markers, their hexadecimal code, and their definition/purpose.

Vulnerability Exploitation – No software is ever completely protected, and it is almost impossible to prevent the presence of vulnerabilities during the development of a large-scale software project. Such vulnerabilities, when exploited, can allow an adversary to obtain higher privileges or divert the normal execution flow to an arbitrary malicious code. In addition, in order to view/parse a JPEG image, a viewer/parser program is required, and these programs may have some vulnerabilities. Many vulnerabilities related to JPEG images have been discovered since it was first published, and there are currently 303 known vulnerabilities (CVE – Common Vulnerabilities and Exposures), and 5,520 known related security issues associated with JPEG images. For example, a recently discovered vulnerability (CVE-2018-6612) may allow a remote attacker to cause a denial-of-service when the victim processes a malicious JPEG file.

Marker Name	Hexadecimal Code	Definition/Purpose
APP$_n$	0xFFE0-0xFFEF	Reserved for **app**lication used
COM	0xFFFE	**Com**ment
DAC	0xFFCC	**D**efine **a**rithmetic **c**onditioning table(s)
DHP	0xFFDE	**D**efine **h**ierarchical **p**rogression
DHT	0xFFC4	**D**efine **H**uffman **t**able(s)
DNL	0xFFDC	**D**efine **n**umber of **l**ines
DQT	0xFFDB	**D**efine **q**uantization **t**able(s)
DRI	0xFFDD	**D**efine **r**estart **i**nterval
EXP	0xFFDF	**Exp**and reference image(s)
JPG	0xFFC8	Reserved for **JPEG** extensions
JPG$_n$	0xFFF0-0xFFFD	Reserved for **JPEG** extensions
RES	0xFF02-0xFFBF	**Res**erved
RST$_m$	0xFFD0-0xFFD7	**Rest**art with modulo 8 counter m
SOF$_n$	0xFFC0-3, 5-7, 9-B, D-F	**S**tart **o**f *Frame*
SOS	0xFFDA	**S**tart **o**f *Scan*
TEM	0xFF01	For **tem**porary use in arithmetic coding
SOI	0xFFD8	**S**tart **o**f **i**mage
EOI	0xFFD9	**E**nd **o**f **i**mage

Table 1: Possible JPEG markers

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 **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**

Fig 1: JPEG file structure in hexadecimal view; the bold bytes are markers

A JPEG image begins with the 0xFFD8 maker (SOI– start of image) which is followed immediately by the 0xFFE0 marker (APP 0). A JPEG image ends with 0xFFD9 (EOI– end of image). Figure 1 presents the hexadecimal view of a sample JPEG image file; the bold bytes are the markers.

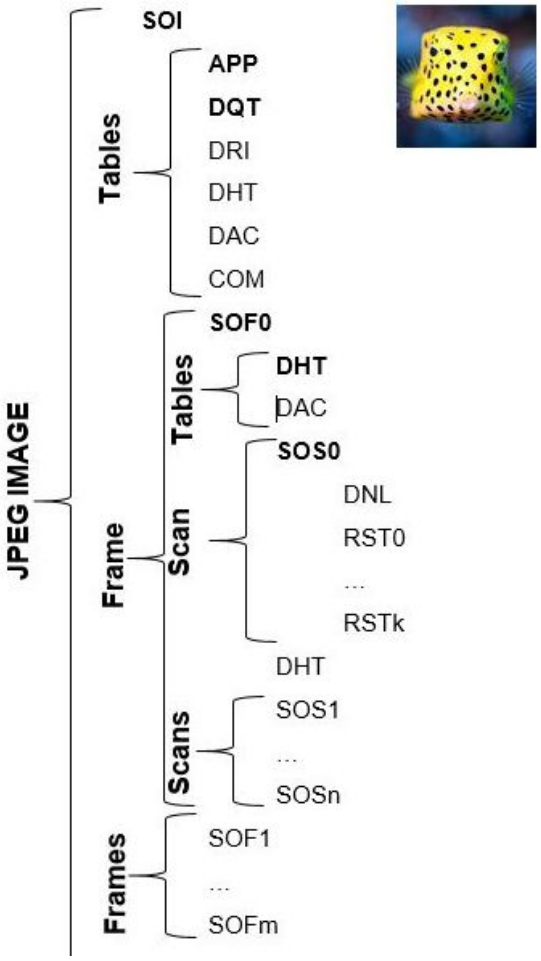

Fig 2: The structure of a simple JPEG image and the hierarchy of the markers and their division into frames and scans. The markers in bold are mandatory or the most common markers.

LITERATURE SURVEY

1. **A. Cohen, N. Nissim, Y. Elovici, "MalJPEG: Machine Learning Based Solution for the Detection of Malicious JPEG Images", 2020**

   This is the paper we have decided to implement for the current project. The paper gives detailed information on how attackers conceal malicious payload in seemingly benign JPEG files and a discriminative list of features for detecting these malicious images. One of the few drawbacks in this is that since we are extracting features from a JPEG file and only a JPEG file, we do not have the means to detect malicious payloads in other file formats. The features detected are exclusive to JPEG files and so cannot be used for other file types.

2. **A. Cohen, N. Nissim, L. Rokach, and Y. Elovici, ''SFEM: Structural feature extraction methodology for the detection of malicious office documents using machine learning methods,'' 2016.**

   This paper gives a detailed study of malicious content in regular office use files like pdfs, doc files, etc., and extracting features that give the highest information gain ratio and then using these to differentiate between malicious and benign using machine learning methods. We have used this paper to try to understand different ways of extracting features from a file. This paper does not contain ways to extract features from JPEG files considering the vast amount of information in such a file.

3. **R. S. Kunwar and P. Sharma, ''Framework to detect malicious codes embedded with JPEG images over social networking sites,'', 2017**

   This paper suggests a framework for finding the presence of code or any kind of extraneous data in a JPEG file. The drawback is that since they are only used to detect whether there is any code/data with the presence of seemingly unwanted information. But that is not always the case. So there is a high chance of false positives without the use of machine learning classifiers.

METHODOLOGY

The MalJPEG machine learning solution works as follows. MalJPEG receives an image as an input. The MalJPEG feature extractor extracts the features proposed into a vector of features.The MalJPEG feature extractor inspects the file statically without actually viewing the image (which requires executing image viewer software that itself might have a vulnerability), and traverses through the JPEG image file structure in order to extract the features. The features are then transferred to a pre-trained machine learning-based model which outputs a classification (benign/malicious) for the input image.

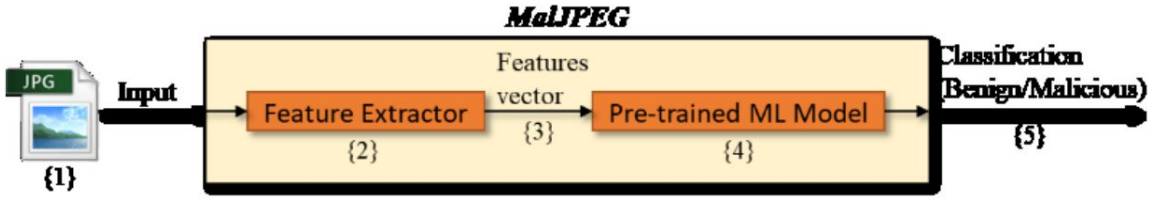

Fig 3: The MalJPEG solution

The MalJPEG features

The features have been engineered after manually examining the structure of many benign and malicious JPEG images. An understanding was gained about how attackers use JPEG images in order to launch attacks and how it affects the JPEG file structure. We also found how malicious JPEG images differ from regular benign JPEG images in terms of file structure. For example, some malicious JPEG files contain data (usually code) after the end-of-file (EOI) marker. In addition, we statistically analyzed the distribution for JPEG markers' frequency and size in both malicious and benign JPEG images and define features that primarily discriminate between benign and malicious JPEG images. The features are very simple, and most of them are based on the presence and size of specific markers within the JPEG image file structure. In addition, the features are relatively easy to extract statically (without actually presenting the image) when parsing the JPEG image file.

#	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

Table 2: The MalJPEG features

Information Gain ranks a feature (attribute) by measuring the reduction in entropy of a given set after dividing it based on a specific feature; this results in the gain of information as a result of using the feature. Specifically, it subtracts the weighted entropies of each subset from the original entropy of the entire set. The entropy characterizes the disorder in an arbitrary set of instances. If the set is completely homogeneous, the entropy is zero; if the set is equally divided, then it has entropy of one. Information Gain assigns a higher rank to features that contribute significantly to discrimination between malicious and benign classes. A malicious JPEG image carries the malicious payload within itself in some way. Therefore, some of the proposed features are indicative of the maximal size of specific markers (e.g., DHT, DQT, COM, APP1, APP12) that are used by attackers to store the malicious payload. The injection of a payload into a marker increases its size beyond the typical size. In some cases, the malicious payload is spread across

multiple markers. Therefore, the reset of the features are indicative of the frequency of specific markers in the file.

An attempt to make a benign file malicious will affect the structure of the image file and thus will be reflected in the selected features. It is important to mention that it is possible that future attack techniques applied on malicious JPEG images will use markers that are not covered by MalJPEG features. Such attacks will likely still affect the structure of the image file and therefore be reflected in the existing selected features which cover the most important markers. In any case, the MalJPEG feature extractor is extendable and can easily be updated to extract new features representative of any other marker in the image file that may be found to be important in the future.

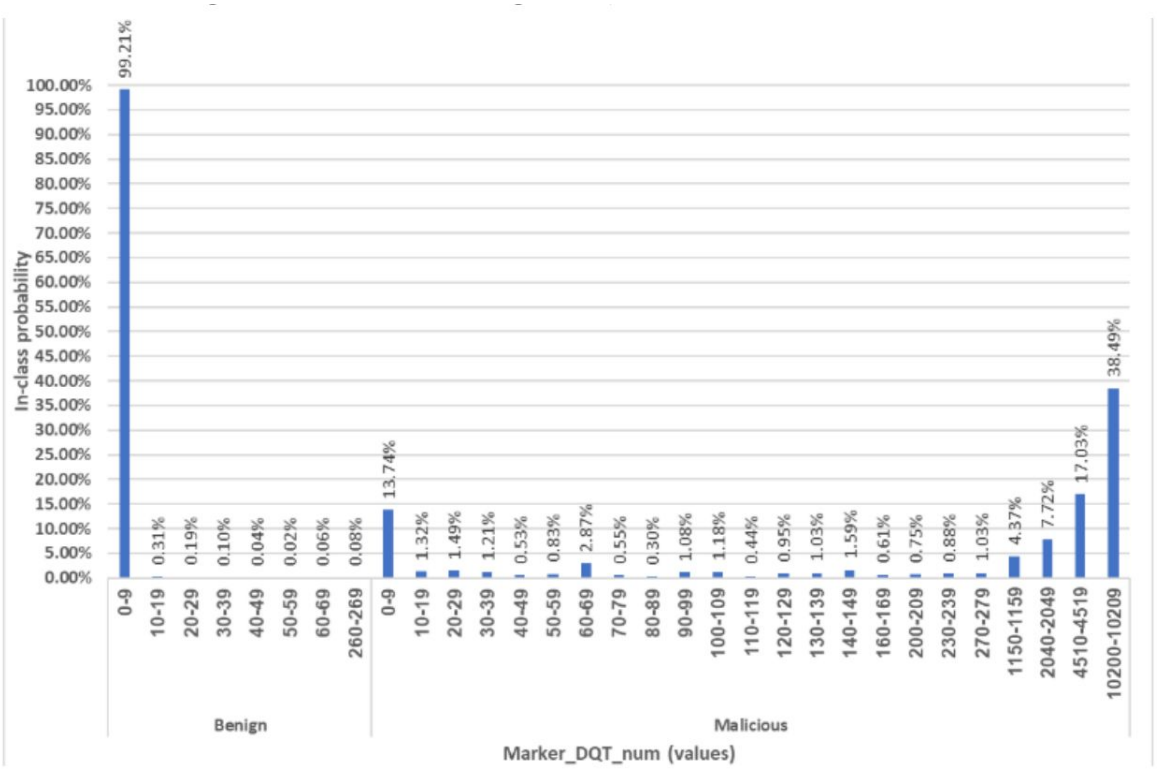

Fig 4: A histogram of the DQT marker values among benign and malicious images.

'Marker_DQT_num' feature (numeric) among the benign and malicious classes. The x-axis represents the number of DQT markers (bin size = 10) in an image, and the y-axis represents the

percentage of images from the class (benign or malicious) that fall in that bin. It can be seen that the distribution is completely different for benign and malicious classes, and this is most noticeable when looking at the range of the values. In addition, 99.21% of the benign images have between zero and nine DQT markers, whereas only 13.74% of the malicious images have between zero and nine DQT markers, and 38.49% of them have between 10,200 and 10,209 DQT markers. In Figure 4 we can see that the DQT marker is much more prevalent in malicious images than in benign images. The DQT marker is, in fact, used by attackers to store the malicious payload. Therefore, the 'Marker_DQT_num' feature is prominent and assists in differentiating between benign and malicious images.

WORK DONE

A. Sourcing of Malicious Images
    - We used the website VirusShare.com to obtain about 110 malicious images.
B. Sourcing of Benign Images
    - We used the website Kaggle.com to obtain more than 6000 benign images.
C. Implementation of MalJPEG feature extractor
    - We wrote a program to extract the ten features mentioned in the paper. The extracted features were added into a .csv file which will be used for training and testing the model.
D. Implementation of MinHash Generic Feature Extractor

The Min-Hash is a technique for quickly estimating how similar two items are. The Min-Hash method generates an N -size signature for a given file based on N simple hash functions . The similarity of two items can be easily computed by calculating the Hamming distance between their signatures; the more matched signatures, the lower the Hamming distance. The signature created by the Min-Hash technique can be used as an input feature for lazy machine learning algorithms which are based on a distance function (e.g., K-Nearest Neighbors).

The Min-Hash method generates a signature based on shingles extracted from the file. A shingle is a fixed length sequence of basic units in the file (e.g., byte, char, etc.). A basic way of extracting shingles is to slide a W -size window, with a stride of S, over the file. N hash functions are applied on each shingle extracted from a file, and the hash results (Long type) are stored on an N -size array. The signature of a file is a vector that contains only the minimal hash value produced by each hash function, across all shingles. The Min-Hash method is very efficient in terms of time and space. Any file, of any size, can be easily converted to an N -size signature.

All programs have been written using Python3.

RESULTS

marker_EOI_content_after_num	marker_DQT_num	marker_DHT_num	file_markers_num	marker_DQT_size_max	marker_DHT_size_max	file_size	marker_COM_size_max	marker_APP1_size_max	marker_APP12_size_max	target
0	2	4	11	67	181	11643	0	0	0	0
0	2	4	11	67	181	11746	0	0	0	0
0	2	4	11	67	181	10460	0	0	0	0
0	2	4	11	67	181	13539	0	0	0	0
0	2	4	11	67	181	11250	0	0	0	0
0	2	4	11	67	181	10226	0	0	0	0
0	2	4	11	67	181	13892	0	0	0	0
0	2	4	11	67	181	10168	0	0	0	0
0	2	4	11	67	181	10828	0	0	0	0
0	2	4	11	67	181	11292	0	0	0	0
0	2	4	11	67	181	12851	0	0	0	0
0	2	4	11	67	181	9637	0	0	0	0
0	2	4	11	67	181	10088	0	0	0	0
0	2	4	11	67	181	9664	0	0	0	0
0	2	4	11	67	181	13268	0	0	0	0
0	2	4	11	67	181	11811	0	0	0	0
0	2	4	11	67	181	8543	0	0	0	0
0	2	4	11	67	181	14974	0	0	0	0
0	2	4	11	67	181	8569	0	0	0	0
0	2	4	11	67	181	11731	0	0	0	0
0	2	4	11	67	181	19215	0	0	0	0
0	2	4	11	67	181	9058	0	0	0	0
0	2	4	11	67	181	9934	0	0	0	0
0	2	4	11	67	181	9612	0	0	0	0
0	2	4	11	67	181	13053	0	0	0	0
0	2	4	11	67	181	8377	0	0	0	0
0	2	4	11	67	181	20038	0	0	0	0
0	2	4	11	67	181	11241	0	0	0	0
0	2	4	11	67	181	16726	0	0	0	0
0	2	4	11	67	181	9517	0	0	0	0
0	2	4	11	67	181	11068	0	0	0	0
0	2	4	11	67	181	9296	0	0	0	0
0	2	4	11	67	181	11111	0	0	0	0
0	2	4	11	67	181	11587	0	0	0	0
0	2	4	11	67	181	12519	0	0	0	0
0	2	4	11	67	181	10166	0	0	0	0
0	2	4	11	67	181	13762	0	0	0	0

Fig 5: The created dataset for benign images with MalJPEG Extractor

	A	B	C	D	E	F	G	H	I	J	K	L
1	marker_EOI_content_after_num	marker_DQT_num	marker_DHT_num	file_markers_num	marker_DQT_size_max	marker_DHT_size_max	file_size	marker_COM_size_max	marker_APP1_size_max	marker_APP12_size_max	target	
2	0	1	1	37	132	418	39781	0	9072	0	120	
3	0	2	4	12	67	58	10311	0	0	0	120	
4	0	2	4	10	67	181	3754	0	0	0	120	
5	4815	2	4	12	67	181	7308	62	0	0	120	
6	0	1	1	9	132	196	2626	0	0	17	120	
7	62753	2	4	12	67	60	69348	12	0	0	120	
8	71524	3	3	129	132	418	168853	0	15025	0	120	
9	0	2	4	12	67	181	329686	59	0	0	120	
10	0	1	1	9	132	85	593	0	0	17	120	
11	0	0	1	9	0	162	23789	37910	0	17	120	
12	58	2	4	12	67	181	52312	59	0	0	120	
13	0	2	4	11	67	72	11275	0	0	0	120	
14	0	1	1	13	132	418	10904	0	4042	0	120	
15	0	1	1	9	132	230	77237	0	0	17	120	
16	0	2	4	11	67	49	1763	0	0	0	120	
17	60	2	4	12	67	181	1965	59	0	0	120	
18	0	1	1	10	132	146	7413	0	877	17	120	
19	0	1	1	10	132	142	2948	0	877	17	120	
20	0	2	4	12	67	181	13168	59	0	0	120	
21	0	2	4	13	67	83	82554	0	221	0	120	
22	0	1	1	9	132	135	1607	0	0	17	120	
23	0	1	1	10	132	218	61805	0	879	17	120	
24	0	2	4	12	67	181	1434	59	0	0	120	
25	0	2	4	11	67	75	10153	0	0	0	120	
26	0	0	1	9	0	162	60889	37910	0	17	120	
27	0	2	4	12	67	181	1613	60	0	0	120	
28	0	0	1	9	0	162	4098	37910	0	17	120	
29	0	1	1	9	132	242	99835	0	0	17	120	
30	0	1	1	10	132	168	10962	0	952	17	120	
31	0	2	10	26	67	55	5363	0	0	0	120	
32	251	2	4	13	67	62	10751	0	811	0	120	
33	21752	2	4	12	67	181	24430	59	0	0	120	
34	0	1	1	18	132	227	3959	0	0	17	120	
35	0	2	4	11	67	88	720501	0	0	0	120	
36	0	0	1	9	0	162	4098	37910	0	17	120	
37	0	1	1	10	132	163	33848	0	793	17	120	
38	0	2	4	12	67	181	3504	0	1028	0	120	

Fig 6: The created dataset for malicious images with MalJPEG Extractor

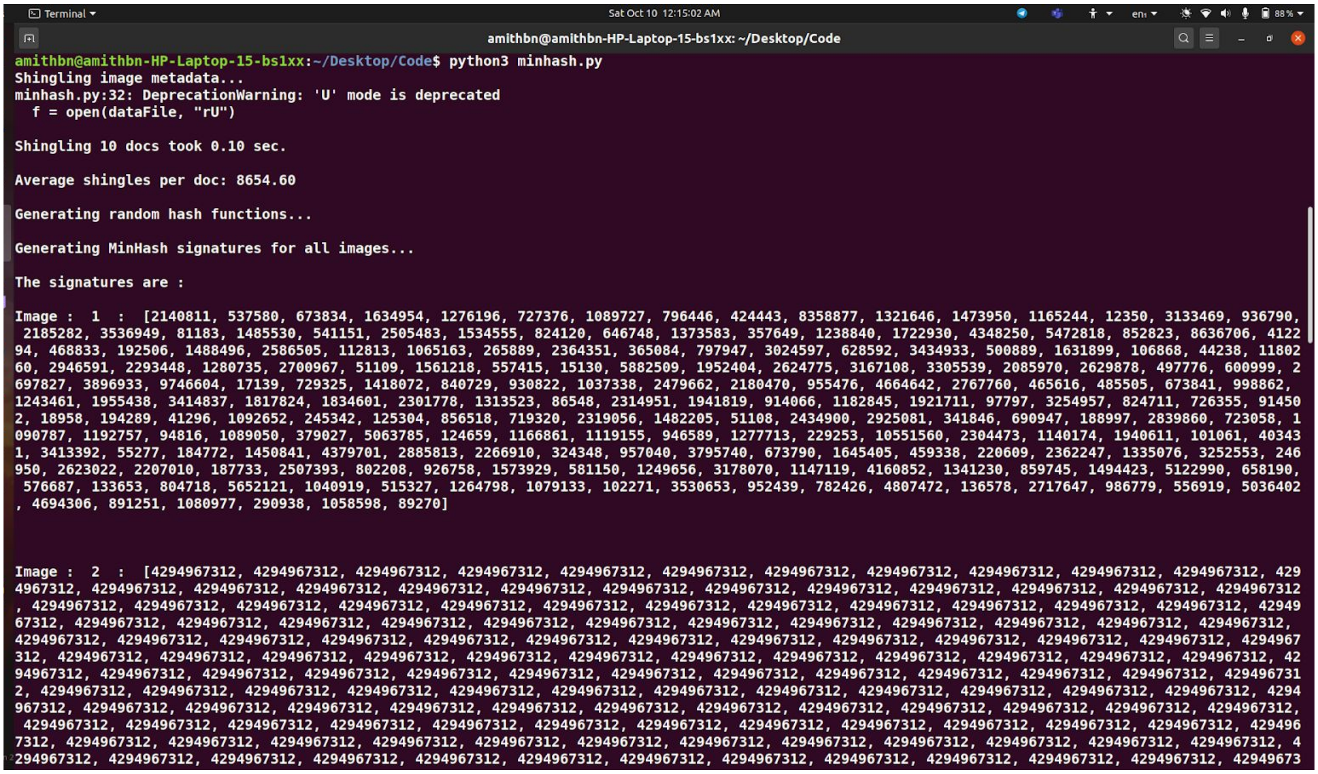

Fig 6: The signature vector generated by the MinHash extractor

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.