Detecting Malware in JPEG Files Through EXIF Tag Analysis using Machine Learning

Partha Majumdar
Bangalore, India

***Abstract*— The paper demonstrates that malware can be detected through supervised machine learning. Using machine learning for malware detection has technical and commercial benefits. This is because malware detection in the current context is about constantly researching for new signatures developed by hackers and programming for detecting these signatures in files. This process is effort intensive and time intensive. The result is an expensive process for malware detection. If the machine could be made to detect malware, not only will the process of malware detection become less expensive due to less human effort involved; the time required to detect new malware will reduce drastically.**

Keywords*— *Malware detection, cyber security, machine learning

I. INTRODUCTION

Cybersecurity is of paramount importance in the modern world for any government and any enterprise. Governments and Enterprises are under constant cyber-attacks from all kinds of adversaries. It is said that the next war will be fought in the Cyber Space. So, it is of utmost importance to develop products and strategies to be able to deal with the threat of cyber-attacks.

One of the means of cyber-attacks is through malwares. Malwares are malicious computer programs embedded in software of regular use. For example, an email may carry a malware which can infect the machine of the receiver of the email and then spread across an enterprise.

One very ubiquitous file exchanged by people across the globe are images. The advent of social media has boosted the exchange of images. Images can be exchanged not only between 2 computers: but also using devices like mobile phone, etc. Also, it is the tendency of people to spread images across wide circles. So, developers of malware target embedding their malicious code into images so that it gets a lot of traction.

One of the very popular forms of image files are JPEG (Joint Photographic Expert Group) files. So, infecting JPEG files has lots of incentives for malware developers. **This paper describes the use supervised machine learning so that computers can detect the malware in JPEG files without having to constantly feed the computer with knowledge of new signatures devised by hackers**.

JPEG Files have many components that can be infected. This paper deals with only one such component and that is the EXIF (Exchangeable Image File Format) Tags. EXIF Tags were introduced into JPEG files in 2010. Through EXIF Tags, meta data about the photograph is stored in the image file. As the data in the EXIF Tags do not carry any information required for rendering the image, tampering with the EXIF Tags does not result in any distortion in the image. So, **a JPEG Files with tampered EXIF Tags is very difficult for users of the image file to detect.** This is the incentive for malware developers to embed their malicious code in the EXIF Tags.

JPEG Files can be infected other than tampering with the EXIF Tags. This experiment is only about detecting infected JPEG Files created by tampering the EXIF Tags in JPEG Files. So, to build a comprehensive product for malware detection in JPEG Files, other components must be researched. However, it has been empirically established that at least 60% of the malware in JPEG Files is introduced through infecting the EXIF Tags.

II. RESULTS OBTAINED FROM THE EXPERIMENT

Before discussing the technical details of the experiment, let us have a look at the results obtained from the experiment.

During training, from a Training Set of 2,719 clean JPEG Files, 2,717 JPEG files got classified a clean JPEG file and 2 clean JPEG Files got classified as JPEG files containing malware. However, from a training set of 358 JPEG files containing malware, none of these files got classified as clean JPEG files. This was a significant result as the most unwanted result of this project was to get a False Positive. A clean JPEG file getting classified as a malware is less dangerous than a malware getting classified as a clean file.

During testing, from a Test Set of 471 clean JPEG files, 450 JPEG files got classified as clean JPEG Files and 21 clean JPEG Files for classified as JPEG Files containing malware. However, from a test set of 48 JPEG files containing malware, none of the files got classified as clean JPEG files.

We see that the training accuracy is 99.935%. And the testing accuracy of 95.953%. From this observation, it is safe to conclude that the model does not suffer from overfitting.

Through training the model with around 3,600 files, the model can detect malware in JPEG files with an accuracy of about 95.9%. However, *the model has a notable weakness that the model does not deal with JPEG files with malware which do not contain any EXIF Tags*.

The project demonstrates that this strategy for malware detection using machine learning can be reliable; and thus, can save a lot of money and effort for detecting malware. The concepts used in this project can be extended to other file types; thus, broadening the utility of this product. *However, it must be stated that this is only part of the solution. There are cases beyond what this project covers*.

III. ABOUT JPEG FILES

JPEG Files are compressed files used to store images. The JPEG Files can be identified by a marker 0xFFD8 at the start of the file.

The information in the JPEG Files can be classified into 2 types - Image Data and EXIF Tags. The Japan Electronic Industries Development Association (JEIDA) produced the initial definition of EXIF in 1995. EXIF Tags were added to JPEG file format in 2010 when EXIF version 2.3 was released.

A. *Image Data*

The **Image Data of JPEG files** have the following segments:¶

- Header
- 2 Quantization Tables
- Frame Information

The **Header** segment contains the following data:

- Identifier
- Version
- Units
- Density
- Thumbnail

One **Quantization Table** contains the data regarding the luminance of the image. It is a 8 * 8 table. Another **Quantization Table** contains the data regarding the chrominance of the image. It is a 8 * 8 table.

The **Frame Information** is a series of Huffman encoded tables containing the bit pattern of the image.

The important thing to note is that if any of above values is tampered with, the JPEG file may not render appropriately. So, the developers of Malware generally do not tamper with this part of JPEG files to introduce malware.

B. *EXIF Tags*

EXIF Tags provide additional information to the JPEG Files. The EXIF Tags can be altered programmatically to alter the nature of the image stored in the JPEG files. For example, by altering the Exif Tag BrightnessValue, the brightness of the image can be altered. Similarly, by altering the EXIF Tags ExifImageHeight and ExifImageWidth, the size of the image can be altered.

EXIF Tags provide the facility for photograph editors to make enhancements/alterations to an image stored as a JPEG File. As these are values which can be altered, the developers of Malware alter these tags to introduce their spurious code. One way to clean an infected JPEG file is to convert the JPEG File to a Bitmap Image (BMP) File. BMP Files can only contain the image information and does not support any EXIF tags. Converting to BMP files gets rid of the EXIF tags and thus the JPEG files gets cleaned of malware.

A JPEG File may contain EXIF Tags or may not contain EXIF Tags.

IV. ABOUT THE DATA

The initial data for this experiment was obtained from C3i Labs of IIT, Kanpur. C3i Labs provided **3,124** clean JPEG files and **904** JPEG files with malware.

As the total number of files were less, **4,001** JPEG files were collected from my library of photographs and from the collection of photographs from my friend. These files were transferred using Google Drive. Transferring using Google Drive was a round of check to ensure that these files were clean as Google Drive discards any JPEG files containing a malware.

As the number of JPEG files containing malware was less, **160** JPEG files containing malware were collected from a friend who works in the area of Cyber Security.

So, the total data set contained **7,125** clean JPEG files and **1,064** JPEG files with malware.

A. *Training Set*¶

Out of the total data set, **6,011** clean JPEG files and **969** JPEG files with malware were used for the training set. Out of these files, **2,719** clean JPEG files and **358** JPEG files with malware had EXIF Tags.

B. *Test Set*

Out of the total data set, **1,114** clean JPEG files and **95** JPEG files with malware were used for the test set. Out of these files, **471** clean JPEG files and **48** JPEG files with malware had EXIF Tags.

V. STRATEGIES USED TO ARRIVE AT THE FINAL MODEL

The initial part of the project involved finding reliable libraries in Python to be able to read the JPEG files. Once PIL library was found and JPEG files were being read, two significant discoveries were that not all the JPEG files in the obtained data set were JPEG Files and that many of the JPEG Files did not contain EXIF tags. Both these reasons reduced the data set. This discovery also led to the following conclusions:

- If the file was not a valid JPEG File, the software would just reject these files.
- If the file was a valid JPEG file but did not contain EXIF tags, then the file would not be considered as part of this experiment.
- Some of the JPEG files containing malware were found to be not JPEG files. This is possible as the file could have got damaged while it was being manipulated. JPEG files are identified as valid or invalid using the PIL library.
- For the JPEG files containing malware without any EXIF Tags, a check was performed by uploading these files to Google Drive. The files got uploaded to Google Drive. However, when these files were tried to be downloaded from Google Drive, Google Drive reported an error. So, we can conclude that **JPEG files can contain malware even if they do not contain EXIF tags**.
- Out of the JPEG Files containing malware, one file was found which Python program refused to read as the image size in the file was beyond what Python allows to read. It cannot be concluded whether this was a file containing a malware.
- Not all the JPEG Files contain the same set of tags.

A. *Strategy 1: Using the length of the tags as features*

Once the tags were extracted from the JPEG Files, a unique list of tags found across all the JPEG files was created. A data frame was formed with each of these tags as columns.

The length of each tag in every JPEG file was determined. If a tag was not available in a JPEG File, the corresponding column in the data frame was assigned a value of zero. So, now there was a data frame containing only numbers. This data frame could be used for machine learning.

As the data frame was imbalanced, **SMOTE (Synthetic Minority Over-sampling Technique) was used to balance the data frame**.

Using this data frame, a Logistic Regression model was developed. The accuracy obtained through this model was about 56%.

Using the same data frame, a Random Forest model was developed. This model performed slightly better at an accuracy of about 58%.

Then, using this data frame, an Artificial Neural Network was tried using Tensor Flow API. This model performed at an accuracy of about 72%.

B. Strategy 2: Forming TF-IDF

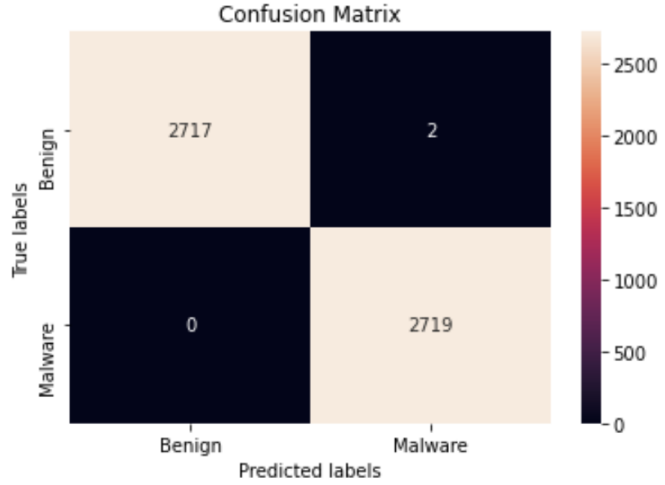
Decided to create TF-IDF (Term Frequency-Inverse Document Frequency) over the tags. A string was formed by concatenating all the tags available in each JPEG file. Then SMOTE was applied to balance the data frame. TF-IDF was formed from these strings. **The TF-IDF provided 45,159 features.**

After forming the TF-IDF, a model was trained using Decision Tree Algorithm. **The Decision Tree Algorithm model gave an accuracy of 98.986% during training.**

Fig 1: Cross Validation Accuracy achieved with Decision Tree Model

```
[25]: print('Start Time: %s' % datetime.datetime.now())

      # Create the Decision Tree Model
      modelDT = DecisionTreeClassifier()

      # Evaluate
      cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
      scores = cross_val_score(modelDT, X, y, scoring='roc_auc', cv=cv, n_jobs=-1)
      print('Mean ROC AUC: %.5f' % scores.mean())

      print('End Time: %s' % datetime.datetime.now())

      Start Time: 2021-06-29 23:55:48.982800
      Mean ROC AUC: 0.98986
      End Time: 2021-06-30 00:00:58.212237
```

Fig 2: Confusion Matrix of Predictions on Training Data using Decision Tree Model

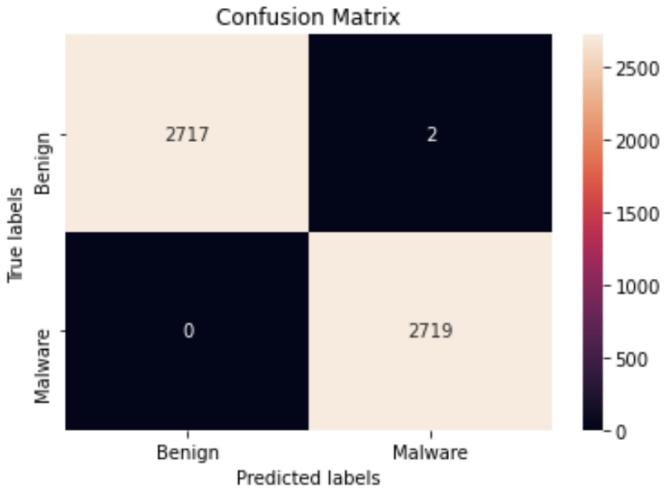


Encouraged by this result, a Random Forest model was created using the same TF-IDF. **The Random Forest Algorithm model gave an accuracy of 99.976% during training**.

Fig 3: Cross Validation Accuracy achieved with Random Forest Model

```
[29]: print('Start Time: %s' % datetime.datetime.now())

      # Create the Random Forest Model
      modelRF = RandomForestClassifier()

      # Evaluate
      cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
      scores = cross_val_score(modelRF, X, y, scoring='roc_auc', cv=cv, n_jobs=-1)
      print('Mean ROC AUC: %.5f' % scores.mean())

      print('End Time: %s' % datetime.datetime.now())

      Start Time: 2021-06-30 00:01:19.820320
      Mean ROC AUC: 0.99976
      End Time: 2021-06-30 00:08:17.370734
```

Fig 4: Confusion Matrix of Predictions on Training Data using Random Forest Model



The test results are provided below.

Fig 5: Confusion Matrix of Predictions on Test Data using Decision Tree Model

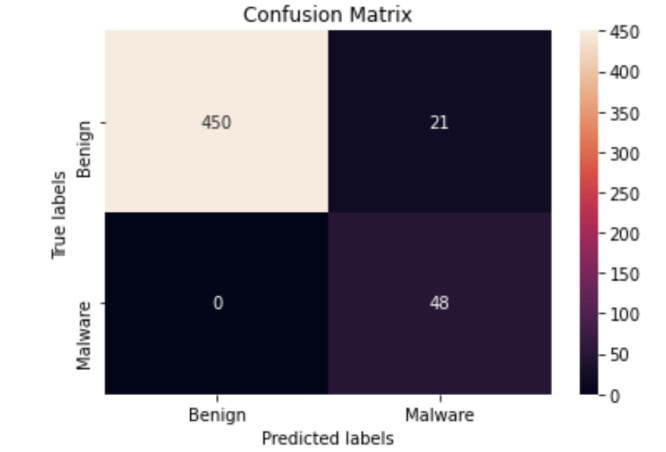
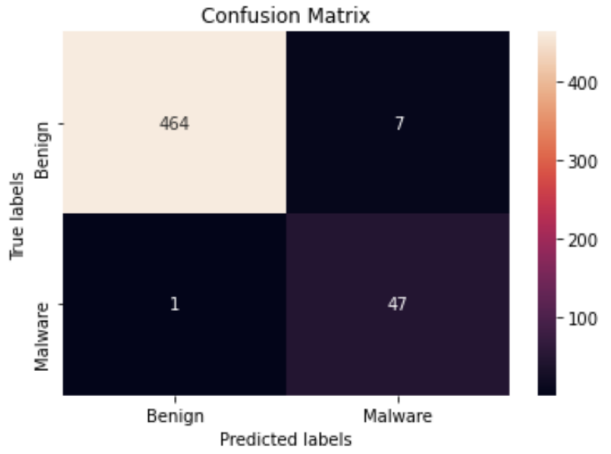


Fig 6: Confusion Matrix of Predictions on Test Data using Random Forest Model



VI. PLANNED NEXT STEPS

2 experiments are planned beyond this experiment.

- To create a classification model using Artificial Neural Network from the features extracted in the TF-IDF. This experiment will be extended to use Convolutional Neural Network (CNN) for classification.
- The TF-IDF formed in this experiment has a weakness. The TF-IDF does not attribute extra weightage to specific identifiers. For example, for embedding a malware in a JPEG file, some form of executable code must be introduced in the code. Generally, executable code would contain identifiers like "eval". Effort will be made to attach extra weightage to such terms while forming the TF-IDF.

ACKNOWLEDGMENT

The author thanks Prof. Anand Handa of IIT, Kanpur for providing the data and for his guidance.

The author thanks Dr. Chetna of Manipal Academy of Higher Education, Dubai for her guidance.

REFERENCES

[1] Hamilton, Eric: JPEG File Interchange Format, Version 1.02 1 September 1992

[2] Saptarshi Bej, Narek Davtyan, Markus Wolfien, Mariam Nassar Olaf Wolkenhauer: "LoRAS: an oversampling approach for imbalanced datasets", 20 August 2019