Detect Image Malware Steganography Using Deep Transfer Learning Model

IYAS ALODAT[1] and MOHAMMAD ALODAT[2]

[1] Jerash University, jerash, Jordan,
eyas.odat@jpu.edu.jo,
WWW home page: http://jpu.edu.jo
[2] Sur College University,Sur,Oman,
Dr.maalodat@suc.edu.om
Sur, Oman

Abstract. The malware would offer the attacker or an un-legitimate user to access the device without authorization as a legitimate user. In this paper, we will discuss how malware hides inside images which can transfer between computers in the background of any system. Also, we will discuss how deep transfer learning will detect malware that hides under images. Lastly, we will make a comparison between deep transfer's models to detect malware in images. We also conclude which model is the best to use in the system to detect malware.

Keywords: Steganography, Cyber Security, Malware, Deep Transfer Learning, Keras

1 Introduction

The crime-as-a-service sector is rapidly developing, making increasingly innovative new developments available to cybercriminals so they can successfully reach their targets. These technologies have become complex threats that can be tailored to the security mechanisms used by consumers and organizations to fight cyber-crime. These are one of the main challenges which is undoubtedly one of malware classifications. Malware that looks "fashionable" today is already outdated tomorrow and is being replaced by others with completely different or improved features. Meanwhile, the latest types of malware continue to coexist with older forms. Therefore, classification in the ecosystem of cybercriminals is extremely complex [5] [6].

In the face of such a context, typical cyber threat intelligence blacklists and indices (IOCs) are not sufficient to deal with the threat. Who changes himself when he finds out that he has been identified. The great power of cybersecurity deep learning is that it makes it possible to learn from this dynamic in real time and develop new classification criteria without human intervention. This allows us to quickly identify whether a person is interacting with their computer or a robot, or if there is a cybercriminal trying to take a user account or interact with a user account from anywhere in the world (remote access to a Trojan horse).

Images can carry a lot of data that is usually invisible to the human eye, as many Facebook users discovered when they examined a partial clip for hidden picture tags attached to users photos. The type of metadata associated with Facebook and Instagram photos is not comparable to the complex methods that threaten the use of craft images that can deliver malicious code or steal user data. Over the past few years, there has been a marked increase in malware campaigns in the wild using the new technique of steganography to hiding-like tricks to embed hidden messages in photos and files.

2 Related Work

Areas most relevant to our proposed work are Android malware hide methods and Android malware detection methods, but they apply a range of available programs. Regardless of the discovery method that those programs carry out, we rely on deep learning techniques and algorithms in our study [2][3][4].

In [16], there is a relationship to what we will do, where a comparison is made between the methods used in artificial intelligence and machine learning for students grades. And also in [15], in terms of what we did there is a similar comparison between the surrounding environment for the leadership process and overcoming human defects.

3 Information's Concealment

Steganography can hide code in plain text, such as inside an image file. This means that messages or information can be hidden inside a non-confidential text as a carrier of these messages and information. In this way, malicious parties use this technology to compromise devices by hosting an image on a website or sending a picture in an e-mail. Hidden data or a carrier file does not have to be images; in fact that digital images are just streams of bytes like any other file which makes them an especially effective way to hide secret text and other data [8]. The science of steganography is a form of obfuscation? that is very different from cryptography, which is the practice of writing encrypted or encrypted messages. The encrypted messages are clearly hiding something, and require specialized decryption methods.

Steganographic letters are like regular letters but cleverly conceal something unexpected. For example in the following statement (He eats like Lucifer only, what other rogue likes Durian!). When we read this message using a familiar technique, it is possible to identify the basic idea behind how to conceal information. The secret message, ”Hello, world” is not encrypted, the reader just needs to know how to look at the message in a certain way to detect it, and we did not have to add any additional data to the ”carrier” in order to transmit it. Although the process of concealment information is much more technical, it is basically the same idea on a lower level.

In previous examples, the human mind is deciphering the hidden message in plain text. But computer programs read bytes, not natural language. It turns out that this makes it possible to hide messages in plain text that are easy for computers to perform and analyze at the same time that it is almost impossible for humans to discover without assistance. In fact, due to the nature of the image file formats, it is possible to hide not only text strings but also the entire files in jpg format, other image formats are also considered in the technology used; this can also be done without inflating the overall file size of the original image.

4 Create Malware Images

To identify images using a deep learning algorithm, we will take images from both benign and malware archives. We are just going to do a binary description (malware and benign class). Multi-class grouping can also be achieved using this method, with the assumption that a variant of malware files would have images distinct from the others. If our dataset is ready, we transform each file to a 256 x 256 gray-scale image (each pixel has a value between 0 and 255) by performing the following steps for each image: First: Read 8 bits at a time from the file. Second: Treat the 8 bits as a binary number and convert it to it's corresponding integer. Third: Enter the number as the pixel value.

A file with a maximum size of 64 KB will fit a 256 x 256 image. If the file size is greater than 64 KB, the remaining contents would be dropped. On the other hand, if the file size is smaller than 64 KB, the remaining images would be padded with 0's. Since the identification of malware is performed in real time, we need to identify the picture as benign or malware within seconds. Keeping the image generation process quick and fast would help us save precious time.

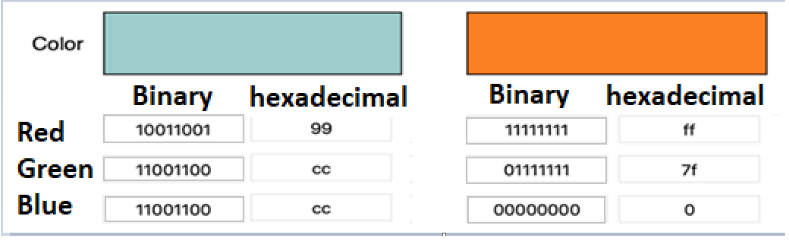
5 Steganography Hides Information

Upon a look at some of the basic ways you can hide text in an image file. One simple way is to simply append the text to the end of the file. This way does not prevent the image from being displayed normally, nor does it change the visual appearance of the image. We simply append "hello world" to the end of the file. The output from hex dump shows us adding the extra bytes.

The plain text string can easily be discarded or read by a program. In this case, we will be using a utility to invert the hex number and print it in plain text. For example, an image that was received displays a picture in an image viewer application normally, but when inspected using the WinRaR archiving utility, we can see that the unpacked .jpg file contains a secret 28 byte text file.

These types of basic approaches can be helpful in collecting user data, but they do have some disadvantages. First, they inflate the file size, and second, they change the file hash. It is still very convenient for security tools to spot because of it's unexpected format.

The best way is to enter into the code at the binary stage and deal with the least important bits (LSBs) of each pixel. Pixels can be represented in a 3-byte color image, one per RGB each (red, green, blue). Suppose we have three bytes representing a particular color as seen in figure 1. You should swap the last four bits of the orange code with the first four bits of the turquoise code, to produce the composite RGB [9].



	Binary	hexadecimal	Binary	hexadecimal
Color	🟦		🟧	
Red	10011001	99	11111111	ff
Green	11001100	cc	01111111	7f
Blue	11001100	cc	00000000	0

Fig. 1. Color with binary to swap least important bits (LSBs).

If we build a program to read and extract these last 4 bits individually, we have essentially concealed the turquoise symbol within the orange color code. Two pixels at one price, as there is no rise in file size. We can relay our secret message without raising the bandwidth of the original message and without manipulating the file format, so there is nothing for simplistic detection methods that rely on searching files to find them. In reality, the code is absolutely crowded before it is regrouped by the attacker.

Shortly, this ensures that an intruder will use the last four bits of encoded RGB data to write additional data without sacrificing the graphic display of the image or amplifying the file size. Hidden data will then be read by another program and used to reconstruct a malicious file or sort user data.

LSB processing is only one of a range of steganography techniques. There are actually a variety of other cases in which photos and other file forms can be manipulated to conceal a hidden message. The attackers have used information concealed in network protocols, the so-called ”network hiding” to relay hidden messages. In both situations, the idea is the same: cover in plain sight by uploading an unseen message to the visible carrier.

Steganography for shielding information has affected both Windows and Mac OS operating systems. An intruder has been found to use cryptography to conceal portions of the ransomware attack code, add malicious JavaScript, and even download encryption software.

6 Experiments Setup Simulation And Dataset

The dataset was dealt with by Hacettepe University's Computer Engineering Multimedia Information Lab. It provides an RGB-based Core Fact Dataset for evaluating vision-based multi-class malware identification studies [1]. We used Keras Framework with TensorFlow in the back-end, this is about deep learning libraries. To manipulate and process our images, we use Pandas and Scikit-learning libraries. All experiments ran using Python 3.7 with notebook IDE.

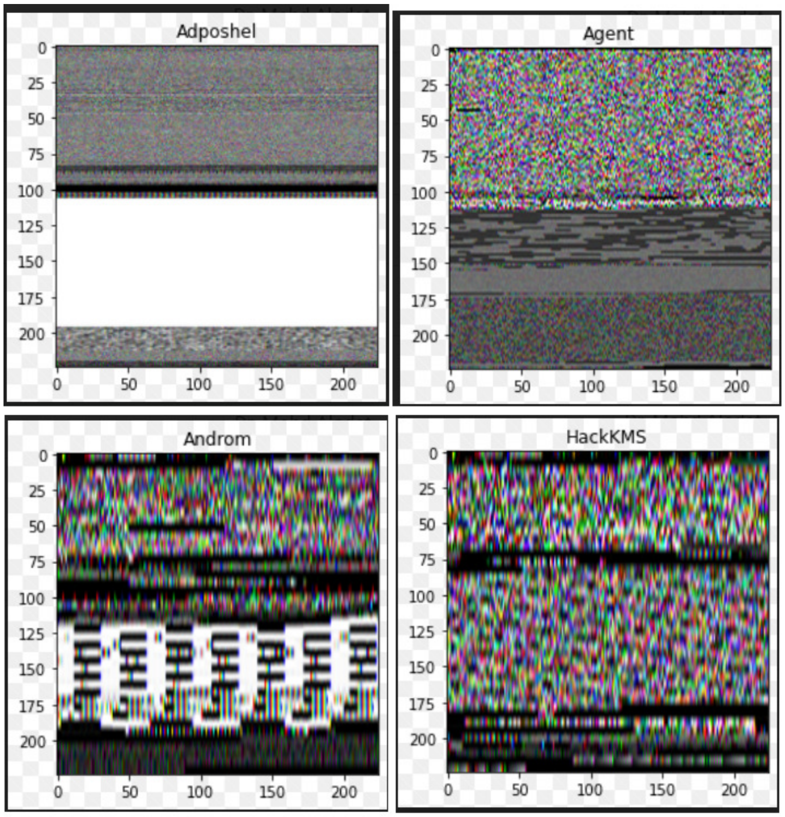


Fig. 2. Malware sample from dataset

A Convolutional Neural Network (CNN, or ConvNet) is a special type of multi-layer neural network designed to distinguish visual patterns directly from pixel images with limited pre-processing [7], in figure 2 we can see samples of malware images. The ImageNet project is a broad graphic library intended to be used in visual object recognition program testing. We used one of the most deep learning libraries called Keras. We simulate 18 kind of malware in different kind of models of deep transfer learning such as MobileNetV2, InceptionV3,ResNet50, LittleVGG [10][11][12][13][14].

7 Performance Evaluation

In order to perform a performance measurement at our work in models, we will use Classification Accuracy, Confusion Matrix and ROC curve. Classification Accuracy is a measure used to classify models, where the number of correct predictions is compared with the total number of predictions provided by each model.

Confusion Matrix is one of the evaluation methods that is done through the result of the model and uses four categories: True positive, which refers to values that are expected to be positive. False positives refer to values that are expected to be positive but come negatively, and therefore they are false, true negative refers to values that are also expected to be negative so they are true. While false negative indicates values that are likely to be negative but are positive, therefore they are false.

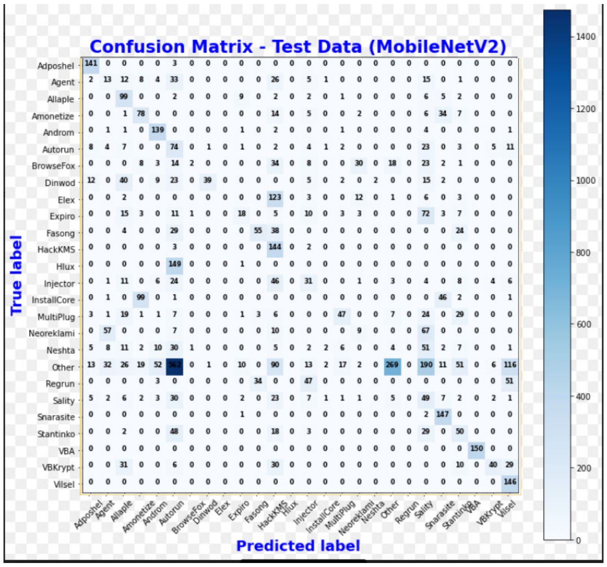
ROC curve is a curve through which we compare two variables, the first is the true positive and the second is the false positive. True positive defines the positive values that were correctly identified by the model as positive. False positive defines the negative values that were also determined to be negative by the model. The plot of the ROC curve will present the true positive rate of a model in relation to it's false positive rate in different places.

8 Results

We describe our models to every DTL (Deep Transfer Learning) by confusion matrix, each figure will explain 26 different types of malware. As we can observe from the list of rates at table 1, we can decide the best model that can predict our malware, the MobileNetV2 was the best choice from the list of rates.

In figure 3, we can also observe from the confusion matrix that MobileNetV2 is the best model because it has a large number of true values which are in the left, and predicted values are in the top. The correct predictions will take place at the diagonal in the matrix.

In our work we have to focus at precision or specificity, this matric can explain false negatives. We need false negatives matrices for non-malware caught by malware filters. In another words, true positives are malware and false negatives are not flag malware. In these ways false negatives it's more acceptable than false positives.



	Adposhel	Agent	Allaple	Amonetize	Androm	Autorun	BrowseFox	Dinwod	Elex	Expiro	Fasong	HackKMS	Hlux	Injector	InstallCore	MultiPlug	Neoreklami	Neshta	Other	Regrun	Sality	Snarasite	Stantinko	VBA	VBKrypt	Vilsel
Adposhel	141	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Agent	2	13	12	8	4	33	0	0	0	0	0	26	0	5	1	0	0	0	0	0	15	0	1	0	0	0
Allaple	0	0	99	0	0	2	0	0	0	9	0	2	0	2	0	1	0	0	0	0	6	5	2	0	0	0
Amonetize	0	0	1	78	0	0	0	0	0	0	0	14	0	5	0	0	2	0	0	0	6	34	7	0	0	0
Androm	0	1	1	0	139	0	0	0	0	1	0	2	0	0	0	1	0	0	0	0	4	0	0	0	0	1
Autorun	8	4	7	0	0	74	0	1	0	1	0	2	0	4	1	2	0	0	0	0	23	0	3	0	5	11
BrowseFox	0	0	0	8	3	14	2	0	0	0	0	34	0	8	0	0	30	0	18	0	23	2	1	0	0	0
Dinwod	12	0	40	0	9	23	0	39	0	0	0	0	0	5	0	2	0	2	0	0	15	2	0	0	0	0
Elex	0	0	2	0	0	0	0	0	0	0	0	123	0	3	0	0	12	0	1	0	6	0	3	0	0	0
Expiro	0	0	15	3	0	11	1	0	0	18	0	5	0	10	0	3	3	0	0	0	72	3	7	0	0	0
Fasong	0	0	4	0	0	29	0	0	0	0	55	38	0	0	0	0	0	0	0	0	0	0	24	0	0	0
HackKMS	0	0	0	0	0	3	0	0	0	0	0	144	0	2	0	0	0	0	0	0	0	0	0	0	0	0
Hlux	0	0	0	0	0	149	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Injector	0	1	11	0	6	24	0	0	0	0	0	46	0	31	0	0	1	0	3	0	4	0	8	0	4	6
InstallCore	0	1	0	99	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	46	2	0	0	1
MultiPlug	3	1	19	1	1	7	0	0	0	1	3	6	0	0	0	47	0	0	7	0	24	0	29	0	0	0
Neoreklami	0	57	0	0	0	7	0	0	0	0	0	10	0	0	0	0	9	0	0	0	67	0	0	0	0	0
Neshta	5	8	11	2	10	30	1	0	0	0	0	5	0	2	2	6	0	0	4	0	51	2	7	0	0	1
Other	13	32	26	19	52	562	0	1	0	10	0	90	0	13	2	17	2	0	269	0	190	11	51	0	6	116
Regrun	0	0	0	0	3	0	0	0	0	0	34	0	0	47	0	0	0	0	0	0	0	0	0	0	0	51
Sality	5	2	6	2	3	30	0	0	0	2	0	23	0	7	1	1	1	0	5	0	49	7	2	0	2	1
Snarasite	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2	147	0	0	0	0
Stantinko	0	0	2	0	0	48	0	0	0	0	0	18	0	3	0	0	0	0	0	0	29	0	50	0	0	0
VBA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	150	0	0
VBKrypt	0	0	31	0	0	6	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	10	0	40	29
Vilsel	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	146

Fig. 3. Confusion matrix for mobileNetV2

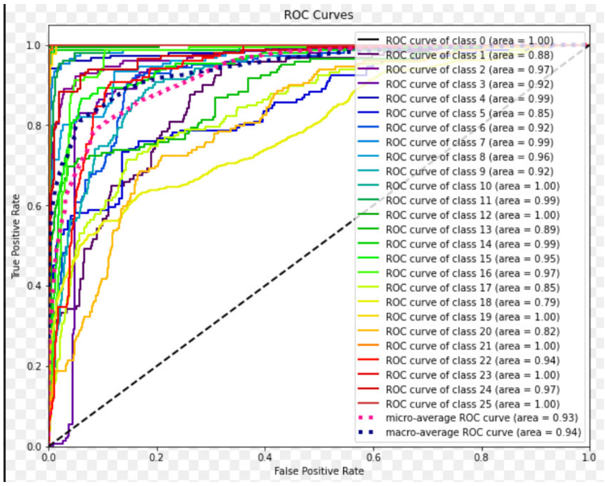


Fig. 4. ROC curve for mobileNetV2

We also made some experiments for each model, as we can see at table 2 the training, validation and testing. At this point we can conclude which model can detect the malware better from another. We can see clearly the best method to detect malware from the confusion matrix is MobileNetV2.

In figure 4, we notice from the ROC curve that it is able to correctly identify the type of malware. The ability to analyze and identify the image correctly, from another side if it was normal.

	Recall	Precision	Score
MobileNetV2	0.917187	0.926349	0.910323
InceptionV3	0.843847	0.834643	0.828221
ResNet50	0.810017	0.800643	0.808221
LittleVGG	0.768251	0.811926	0.775191

Table 1. List of rates computed from a confusion matrix.

	Training	Cross-val	Testing
MobileNetV2	0.94819713	0.94819713	0.95199621
InceptionV3	0.84456111	0.84809954	0.88111015
ResNet50	0.83881235	0.82547898	0.83109547
LittleVGG	0.90258367	0.89804627	0.92458974

Table 2. Data Accuracy after transfer learning— Performance Metrics.

After train simulation, we used freeze technology that freezes the first layer while the simulator is running selects a number of times (After Freeze Epochs=100), and decides the first layer each time. Using this technique our results improved in terms of accuracy, we observed after freezing the accuracy change from 0.94 to 0.96.

9 Conclusion

Many photos that are transmitted by e-mail and with social media, may be hiding a specific threat. Through the analysis that we have done in this study, we can reduce the dangers of malware hidden behind those images, using artificial intelligence techniques and machine learning. MobileNetV2 is the best tool to detect malware in our dataset experiment. We tested and trained the data as shown in the table 2, which was done by Keras. We showed the best results found in the confusion matrix and the ROC curve diagram figure (2, 3).

We were able to reduce these risks very well, and also reduce privacy violations by discovering these types of deceptive users through temptation and carrots that the hacker uses for this purpose to hide in images.

References

1. Bozkir, Ahmet Selman, Ahmet Ogulcan Cankaya, and Murat Aydos. ”Utilization and Comparision of Convolutional Neural Networks in Malware Recognition.” 2019 27th Signal Processing and Communications Applications Conference (SIU). IEEE, 2019.
2. Dini, Gianluca, et al. ”MADAM: a multi-level anomaly detector for android malware.” International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security. Springer, Berlin, Heidelberg, 2012.
3. Aprville, A., and Ange Albertini. ”Hide android applications in images.” Black Hat Europe (2014).
4. Bak, Patryk, et al. ”Application of perfectly undetectable network steganography method for malware hidden communication.” 2018 4th International Conference on Frontiers of Signal Processing (ICFSP). IEEE, 2018.
5. Chandrasekhar, A. M., and K. Raghuveer. ”Intrusion detection technique by using k-means, fuzzy neural network and SVM classifiers.” 2013 International Conference on Computer Communication and Informatics. IEEE, 2013.
6. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. ”Imagenet classification with deep convolutional neural networks.” Communications of the ACM 60.6 (2017): 84-90.
7. Xiao, Yihan, et al. ”An intrusion detection model based on feature reduction and convolutional neural networks.” IEEE Access 7 (2019): 42210-42219.
8. A Al-Juaid, Nouf, Adnan A Gutub, and Esam A Khan. ”Enhancing PC data security via combining RSA cryptography and video based steganography.” (2018).
9. Islam, Md Rashedul, et al. ”An efficient filtering based approach improving LSB image steganography using status bit along with AES cryptography.” 2014 International Conference on Informatics, Electronics & Vision (ICIEV). IEEE, 2014.
10. Ahmim, Ahmed, et al. ”A novel hierarchical intrusion detection system based on decision tree and rules-based models.” 2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS). IEEE, 2019.
11. Vinayakumar, Ravi, et al. ”Deep learning approach for intelligent intrusion detection system.” IEEE Access 7 (2019): 41525-41550.
12. Riyaz, B., and Sannasi Ganapathy. ”A deep learning approach for effective intrusion detection in wireless networks using CNN.” Soft Computing (2020): 1-14.
13. Yang, Liu, Steve Hanneke, and Jaime Carbonell. ”A theory of transfer learning with applications to active learning.” Machine learning 90.2 (2013): 161-189.
14. Tan, Chuanqi, et al. ”A survey on deep transfer learning.” International conference on artificial neural networks. Springer, Cham, 2018.
15. Alodat, Mohammad, and Iyas Abdullah. ”Surveillance Rapid Detection of Signs of Traffic Services in Real Time.” Journal of Telecommunication, Electronic and Computer Engineering (JTEC) 10.2-4 (2018): 193-196.
16. Alodat, Mohammad. ”Predicting Student Final Score Using Deep Learning.” Advances in Computer, Communication and Computational Sciences. Springer, Singapore, 2020. 429-436.