Malware Analysis using Deep Learning
Team : Team Oversight

Naman Rajput(21111043), Nimit Jain(21111045), Prashant Mishra(21111049),
Ravi Shankar Das (21111052), Sahil Shukla (21111054)
{namanraj21, nimitj21, prashantm21, ravisd21, sahils21}@iitk.ac.in

Indian Institute of Technology Kanpur (IIT Kanpur)

April 2022

1 Keywords

Malware Analysis, Deep Learning, Convolution Neural Network, Bi-directional LSTM, Cuckoo Sandbox, Markov Chain based Images, Image representation of malware

2 Executive Summary

2.1 Problem

Recently, there has been a huge rise in malware growth, which creates a significant security threat to organizations and individuals. Despite the continuous efforts of cyber security researchers to defend against malware threats, malware developers discover new ways to evade these defense techniques. Typical machine learning approaches that train a classifier based on handcrafted features are also not sufficiently potent against these evasive techniques and require more efforts due to feature-engineering.

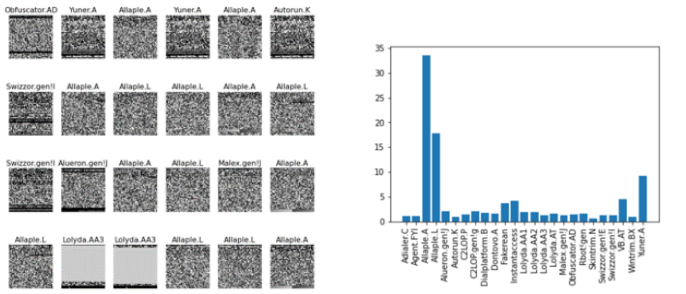
2.2 Goals

In this project we aim to develop a system through which any user can give a PE binary files as input and we will extract the features from it using the dynamic or static analysis and after that we will use our developed models to predict whether the given file is benign or malware. This is going to be very useful in finding malware and taking the required action for securing their system.

2.3 Data Sources

For our project we have used the following datasets:

1. Malimg: This dataset consists of around 10K samples of different types of malware. We resized all the images to 64 x 64 due to architectural design choices. Weighted Cross Entropy Loss is used to handle the class imbalance in training data. The Malimg dataset consists of images, and hence these samples require no pre-processing before applying image-based analysis. However, the binaries corresponding to the Malimg images are not readily available.
   This dataset contains 9339 malware images, organized in 25 families of malware. These 25 families are mentioned below:
   'Adialer.C', 'Agent.FYI', 'Allaple.A','Allaple.L', 'Alueron.gen!J', 'Autorun.K', 'C2LOP.P','C2LOP.gen!g', 'Dialplatform.B', 'Dontovo.A', 'Fakerean', 'Instantaccess', 'Lolyda.AA1', 'Lolyda.AA2', 'Lolyda.AA3', 'Lolyda.AT', 'Malex.gen!J', 'Obfuscator.AD', 'Rbot!gen', 'Skintrim.N', 'Swizzor.gen!E', 'Swizzor.gen!I', 'VB.AT', 'Wintrim.BX', 'Yuner.A'
   The below images represents the characteristics of malimg dataset also build the representative images for each of these classe:

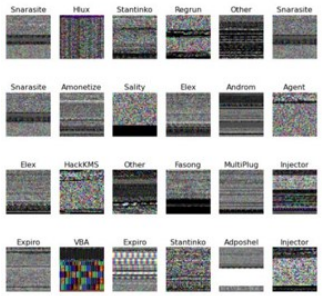


(a) Class representaion for each malware

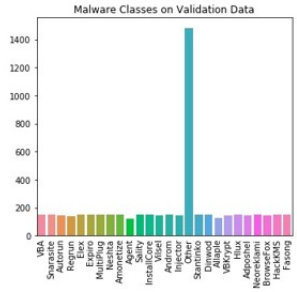
(b) Distribution of multiple classes of malware

Figure 1: Malimg

2. Malevis: This dataset consists of RGB based ground truth dataset to evaluate vision based multi-class malware recognition. MaleVis dataset involves totally 9100 training and 5126 validation RGB images.All the training classes involve 350 image samples while validation set have various number of images. Since malware detection is based on discriminating legitimate ones from the malware, dataset provides a larger set for legitimate samples during validation.
   The below images represents the characteristics of malevis dataset also build the representative images for each of these classes:



(a) Class representaion for each malware



(b) Distribution of multiple classes of malware

Figure 2: Malevis

3. Feature Dataset: Cuckoo, an open-source software, is used to run the PE files in a sandboxed environment and gather execution logs.The execution logs generated contain detailed runtime information of the PE files. A feature dataset contains extracted features from cuckoo's raw execution logs efficiently.There are total 57786 files out of which 30712 files are Benign and 27074 files are Malicious.
This is a sample feature dataset:

(a) Cuckoo generated report in JSON Format

Figure 3: Feature Dataset

2.4 Goals Achieved

1. In our first experiment we have successfully converted PE binary files(byte) data to images using a python script. Then these images served to train a CNN model. This CNN model consists of two 2D convolution layers and three dense layers. This model predicts the malware class for any malware input that is provided in form of PE binary files or images.

2. In our second experiment we have used the markov chain-based images, this approach classifies Autorun.K and Yuner.A properly, so from this experiment we were able to solve the problem that we faced in experiment 2 with the use of markov images.

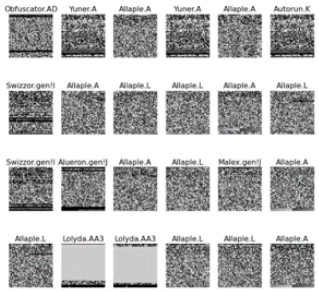
3. In third experiment we were successfully able to extract features using cuckoo sandbox and apply deep learning models to predict byte file as malicious or benign.

4. We have also created a user friendly UI that integrates all the above experiments and user can select any of these three models and evaluate the binary file using that model.

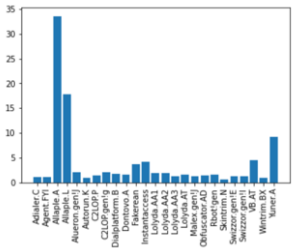
2.5 Experiment validation

1. Maling :

   - Experiment using 64x64 image size
     (a) Experiment 1 validation accuracy: 0.9358
     (b) Experiment 2 validation accuracy: 0.9534



(a) Class representaion for each malware



(b) Distribution of multiple classes of malware

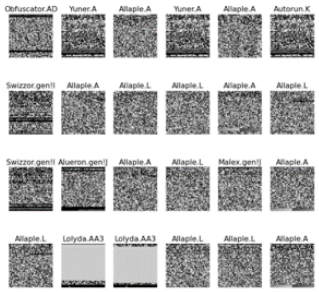
Figure 4: Malimg

2. Malevis :

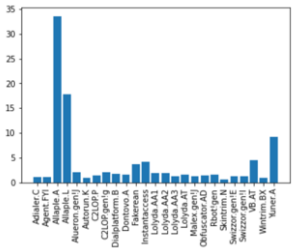
   - Experiments using 64x64 image size
     (a) Experiment 1 validation accuracy: 0.7594
   - Experiments using 128x128 image size
     (a) Experiment 1 validation accuracy: 0.7590

3. Feature Dataset :

   (a) Experiment 3 validation accuracy: 0.9356



(a) Class representaion for each malware



(b) Distribution of multiple classes of malware

Figure 5: Malimg

3 Detailed Explanation

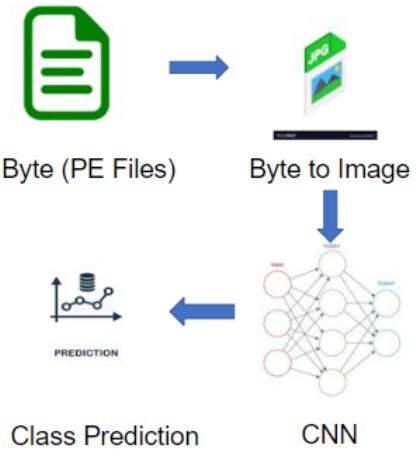
3.1 Problem

For this project we will be using PE binary files as input to deep learning models and after training the model we will be extracting the features from the image and will classify them into different classes. We will be trying various experiments in order to improve the accuracy of the classification tasks. Also we are trying to provide a useful user friendly interface through which the user can know about the byte files being malware or benign.

3.2 Solution Strategy

In static analysis when we tried experiment 1 by training CNN in byte files we came across a problem that our train model is miss classifying between two classes(Yunes.A and Autorun.K) whose samples are less in number. Also to handle the imbalanced dataset we made use of weighted cross enotropy loss. So we considered markov chain based images as an input in experiment 2 and trained the same CNN model and got the previous problem fixed. So for static analysis our experiment is the proposed solution i.e., generating markov chain based images from byte files and then training CNN , after training , making prediction.
In dynamic analysis we generated feature reports of byte files with the help of cuccko sandbox. We extracted the important features from the generated reports and then applied deep learning models consisting of BiLSTM to classify them as malware or benign.
For the end user we have created a UI which is user friendly. So any user can just input their code as byte files and our trained models will tell the user if it is malware or not.

3.3 Architecture

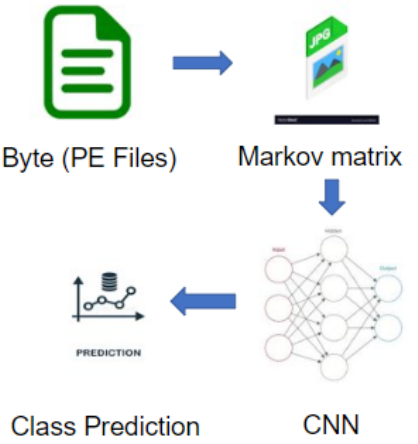


Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 30)	840
max_pooling2d (MaxPooling2D)	(None, 31, 31, 30)	0
conv2d_1 (Conv2D)	(None, 29, 29, 15)	4065
max_pooling2d_1 (MaxPooling2	(None, 14, 14, 15)	0
dropout (Dropout)	(None, 14, 14, 15)	0
flatten (Flatten)	(None, 2940)	0
dense (Dense)	(None, 128)	376448
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 50)	6450
dense_2 (Dense)	(None, 25)	1275

Total params: 389,078
Trainable params: 389,078
Non-trainable params: 0

Figure 6: Experiment1 Architecture



Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 254, 254, 30)	300
max_pooling2d (MaxPooling2D)	(None, 127, 127, 30)	0
conv2d_1 (Conv2D)	(None, 125, 125, 15)	4065
max_pooling2d_1 (MaxPooling2	(None, 62, 62, 15)	0
dropout (Dropout)	(None, 62, 62, 15)	0
flatten (Flatten)	(None, 57660)	0
dense (Dense)	(None, 128)	7380608
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 50)	6450
dense_2 (Dense)	(None, 25)	1275

Total params: 7,392,698
Trainable params: 7,392,698
Non-trainable params: 0

Figure 7: Experiment2 Architecture

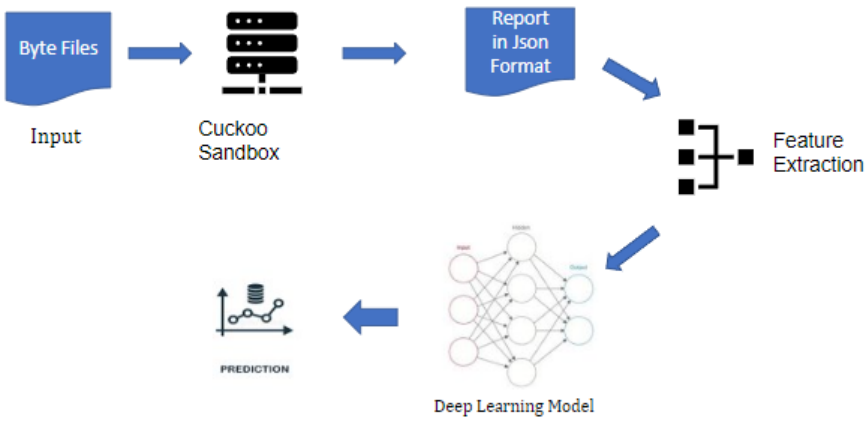


Figure 8: Experiment3(Dynamic Analysis) Architecture

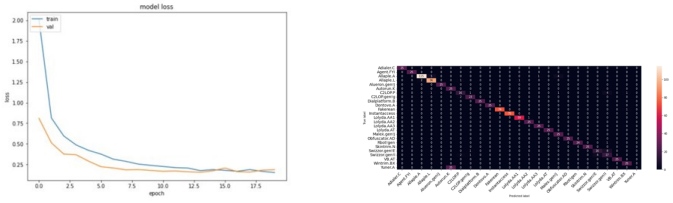
3.4 Experimental Setup

We have created a user friendly UI which was created using python, Javascript in the backend and HTML, CSS and bootsrap on the frontend. When the user selects one of the options the click sends an AJAX request to the development server which was created using FLASK. Flask is a micro web framework written in Python. It is classified as a micro framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.
Another technology used is AJAX, Ajax is a set of web development techniques that uses various web technologies on the client-side to create asynchronous web applications. With Ajax, web applications can send and retrieve data from a server asynchronously without interfering with the display and behaviour of the existing page.
For byte file anaysis and feature generation we have used cuckoo which is an advanced, extremely modular, and open source automated malware analysis system with infinite application opportunities.

3.5 Experimental Results

Experiment 1 resulted in following results:

1. The model converges after running for 10 epochs
2. Overall test Accuracy is coming to be **93.58%**
3. Our model is getting confused between two classed namely Yunes.A and Autorun.K as number of samples were very less

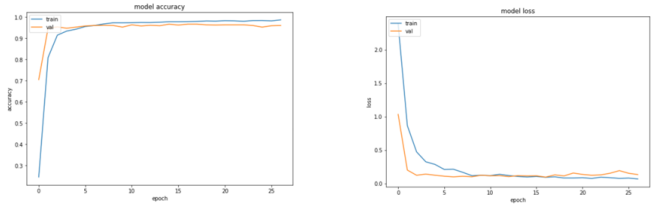


(a) validation loss (b) confusion matrix

Figure 9: Malimg

Experiment 2 resulted in following results:

1. The model converges after running for 12-15 epochs
2. Overall test accuracy is coming to be **95.34%** @5 epoch
3. Our model acheived maximum test Accuracy of **98.92%** @ 17 epoch
4. Also in this experiment by using markov images our model is no more confused between Yunes.A and Autorun.K classes
5. But using this method our model fails to classify Swizzer.gen!E and Swizzer.gen!I (since they came from really close families)

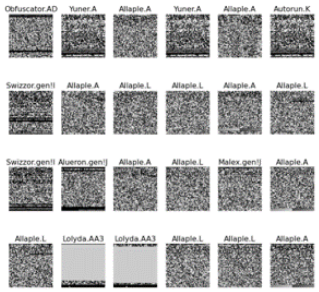


(a) loss (b) Accuracy

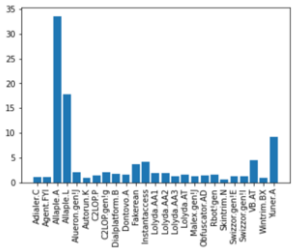
Figure 10: Malimg

In dynamic analysis we have extracted features using cuckoo sandbox and apply deep learning model to predict byte file as malicious or benign. Using above approach we have found out following results:

1. Accuracy of this experiment turns out to be **93.56%**
2. False positive rate for this approach is **0.09%**
3. True positive rate in this experiment is **78.69%**
4. Area under the ROC curve for this experiment is **0.987**



(a) Class representaion for each malware



(b) Distribution of multiple classes of malware

Figure 11: Malimg

4 Conclusion

As we mentioned in our problem statement that we will be extracting features from byte files and classifying them into different classed using different techniques, we were able to achieve our goals, some of them are listed below:

1. Firstly, we have done malware analysis by converting byte files to image files and these image files then fed into the CNN model which predicts the malware class. This approach gives validation accuracy to 93.58

2. In first approach most of the malware were well classified, Autorun.K is always mistaken for Yuner.A. This is probably due to the fact that we have very few samples of Autorun.K in our training set.

3. After that we have implied the markov chain-based images to CNN model and this outperforms the previous approach and increases accuracy to 98.92

4. Second approach classifies Autorun.K and Yuner.A properly, so from this experiment we were able to solve the problem that we faced in experiment 2 with the use of markov images

5. Moreover, Swizzor.gen!E is often mistaken with Swizzor.gen!I, which can be explained by the fact that they come from really close famillies and thus could have similarities in their code.

6. we were also able to integrate our model to our UI through which user can give byte file as input and can use any of the three models to predict whether the file is benign or a malware

7. In third approach we have extracted features using cuckoo sandbox and applied deep learning model to predict byte file as malicious or benign.

4.1 Applicability

Through our UI any user can know upload their byte files and can know their system is affected by a malware or not. If is benign then there is no need to worry about it but if it is a malware then the user can take steps to remove the malware from their system. Also they can take the preventive measures so that next time their system is less vulnerable.

4.2 Future Work

We can perform analysis on the network data and can use the above mentioned experiments to classify them. Also we can deploy our system on a server for real time classification of the uploaded byte files.

References

[1] Zhang, Zhaoqi, Panpan Qi, and Wei Wang. "Dynamic malware analysis with feature engineering and feature learning." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 34. No. 01. 2020.

[2] Yuan, Baoguo, et al. "Byte-level malware classification based on markov images and deep learning." Computers Security 92 (2020): 101740.

[3] Hemalatha, Jeyaprakash, et al. "An efficient densenet-based deep learning model for malware detection." Entropy 23.3 (2021): 344

[4] Huang, Gao, et al. "Densely connected convolutional networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.