Defender

Deep artificial neural network based malware detection

University of Applied Sciences Cologne

B. Eng. Valentin Dederer
Wesseling, Germany
mail@valentin-dederer.de

B. Eng. Jahn Kohlhas
Neuwied, Germany
jahn@kohlhas.info

supervised by:
M. Sc. Jan Bollenbacher
Cologne, Germany

I. Introduction

This project introduces the idea of a deep artificial neural network based malicious file (malware) detection system. Especially in a working environment it is highly important to keep your data save and therefore to keep being operational. Malicious files are often distributed alongside promising mails so that the average office worker falls for it and gets infected by the malicious file which renders him unable to do his work. Furthermore, that action could infect the enclosing ecosystem and do even more harm. On the other hand one cannot simple deactivate all mail attachments or generate too much false positive recognitions which get blocked because that would make the office worker unproductive. Therefore, it is important to implement an effective malicious file detection system.

II. Idea

This project tries to solve a problem that occurs since a long time in traditional malware detection engines. Currently they are based on the use of unique signatures manually chosen by people to identify the presence of malicious code while trying to ensure that there will be no false positives. There are many problems with this approach. First of all it is usually easy to bypass the detection because in dependence of the type of chosen signature, changing a single bit or a few bytes in malicious code may change the signature and therefore make the malware undetectable again. Also it does not scale well as large numbers of people are required to do the manual reverse engineering job. Because of this a whole new industry has risen producing so called *Crypters* whose only job is it to rewrite code so that it is no longer recognized by traditional antivirus systems but still functioning.

Our goal in this project is to teach an artificial neural network to detect windows malware without relying on any explicit signature database we would need to create and keep updated by ourselves. The neural network should obtain by itself what we want to be able to detect and with this reliably detect malicious files. Our only knowledge while training the network is which files are malicious and which are not, but not what part or code makes them malicious. The neural network has to take care of the detection by itself. In order to do this we gathered a dataset of about 5000 files separated in malicious and safe categories.

III. Implementation

Gathering and preprocessing required data for training this network was one of the hardest and time-consuming parts in this project. After looking around a lot of hours searching for available resources, we decided to concentrate on Windows PE *.exe* files, because there most of the malicious files are available. After we decided to just support this type of files it got a bit easier to find sources for malware, but some of them had the problem to be not real malware. To solve this we filtered all grepped malware files with real antivirus detection software and only included real hard malware in our database.

Parsing gathered files was done by a preprocessor written in python. We collected different aspects of the files like used libraries, specific PE file sections, exception handling, digital signature, Shannon entropy, ratio of disk vs. memory size and much more. Then we encoded the extracted features and wrote them to a JSON formatted database file, so that we do not need to extract the features again, if we want to train the network a second time.

For laying out the structure of a neural network which detects given malware the idea was to use a simple neural network with just fully connected (dense) layers. The use of other layers than just dense layers did not seem as a good idea for starting because the features defined as input for the network do not have a locality property. Instead, each feature is independent and could be rearranged in the order of all features.

The results of the experiments were not as expected. The firstly used small dense network with 4 layers produced results which were already very high for such a small network with a validation accuracy of around 92% and a training accuracy of 99.5%. This is a very good result which means that using a dense network was the right choice.

Based on that footing a deep dense network with 9 dense layers was used in the next training with the goal to improve the already good results, but against the expectations the network performed worse on testing data than the previous small dense network even though the validation accuracy improved from about 92% in the previous network to about 94.5% and the training

accuracy stayed at 99.5%.
Because of that missing improvement when using a deeper network the small dense network was used as the starting point of the next experiment, where the goal was to achieve a decrement of network size. That experiment showed that a simple 2 dense layer network with 1024 and 1 neurons was sufficient to keep the results at the same high values without the need for a deeper network.
After that the last experiment block was scheduled in which several options for activation functions, optimizers and neuron counts were tested. All these tests did not yield a better performing network but adding a dropout layer did eventuate in a better performance. All in all the experiments did demonstrate the sometimes not intuitive behavior of neural networks. For example using more layers or more neurons in a layer does not always yield a better performing network. Moreover, it was once more confirmed that the optimizer Adam and the activation function ReLU do produce the best performing network without the need for an elaborate tuning.
All in all it was shown, that using an artificial neural network for detecting malicious files does yield good results. Moreover it was noted, that even a very simple network with just 2 dense layers and 1 dropout layer is sufficient to achieve a detection rate of 96.61% right predicted files while having just 0.97% false positives and 2.42% false negatives.

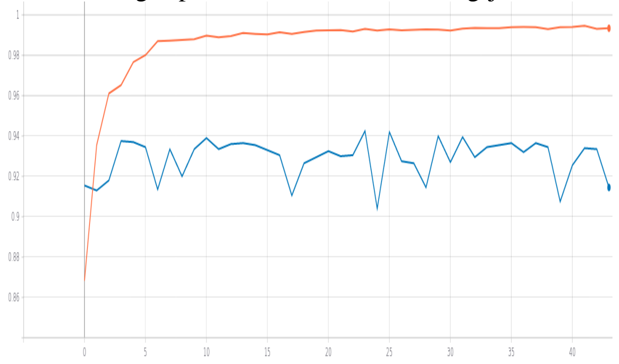

Fig. 1: Training results of reduced dense network

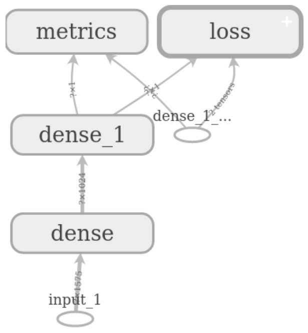

Fig. 2: Topology of reduced dense network

REFERENCES

[1] Zhaoqi Zhang, Panpan Qi, Wei Wang; Dynamic Malware Analysis with Feature Engineering and Feature Learning https://arxiv.org/abs/1907.07352
[2] Shuqiang Lu, Lingyun Ying, Wenjie Lin, Yu Wang, Meining Nie, Kaiwen Shen, Lu Liu, Haixin Duan; New Era of Deeplearning-Based Malware Intrusion Detection: The Malware Detection and Prediction Based On Deep Learning https://arxiv.org/abs/1907.08356
[3] Irina Baptista, Stavros Shiaeles, Nicholas Kolokotronis; A Novel Malware Detection System Based On Machine Learning and Binary Visualization https://arxiv.org/abs/1904.00859
[4] Niket Bhodia, Pratikkumar Prajapati, Fabio Di Troia, Mark Stamp; Transfer Learning for Image-Based Malware Classification https://arxiv.org/abs/1903.11551
[5] 2015, Yuval Nativ, Lahad Ludar, 5fingers https://github.com/ytisf/theZoo
[6] Romain Thomas, LIEF - Library to Instrument Executable Formats, https://lief.quarkslab.com/
[7] Microsoft Corporation https://docs.microsoft.com/en-us/windows/win32/debug/pe-format#the-load-configuration-structure-image-only
[8] Microsoft Corporation https://docs.microsoft.com/en-us/windows/win32/debug/pe-format#the-debug-section
[9] Microsoft Corporation https://docs.microsoft.com/en-us/windows/win32/debug/pe-format#the-tls-section
[10] João Garcia; https://www.virusign.com