Indian Institute of Technology, Kanpur
Automated Malware Classification using Deep Learning

Group Name: AccessDenied

Arpan Kapoor
21111016

Chabil Kansal
21111022

Kaustubh Verma
190424

Manish Kumar
21111037

Vikas Lodhi
21111068

Punyabrat Kalwar
21211403

Under supervision of Dr. Sandeep Kumar Shukla

Keywords: 1D Convolutional Neural Network, Malware Analysis, Linux Malware Autodetection, Malware classification

1 Executive Summary

1.1 Goals

- Malware Detection Application (Command Line Application) using Deep learning to classify binary executable files
- The application should be able to scan an executable just before its execution giving the probability of the executable to fall into some particular malware class
- If the probabilities of being malware is below a set threshold for all the classes then it is said to be benign, else the executable is classified as the malware class with the highest probability.
- Generating a whitelist of executables which were successfully scanned as benign by the model.
- For Windows, developing a GUI application which will take an executable file and predict the probability of it belonging to some malware classes.
- Creating a quarantine location and moving the file into quarantine location if found malicious.

1.2 Data source

- Microsoft Malware Classification Challenge (BIG 15) dataset with 10384 samples belonging to 9 malware classes: `https://www.kaggle.com/c/malware-classification/data`

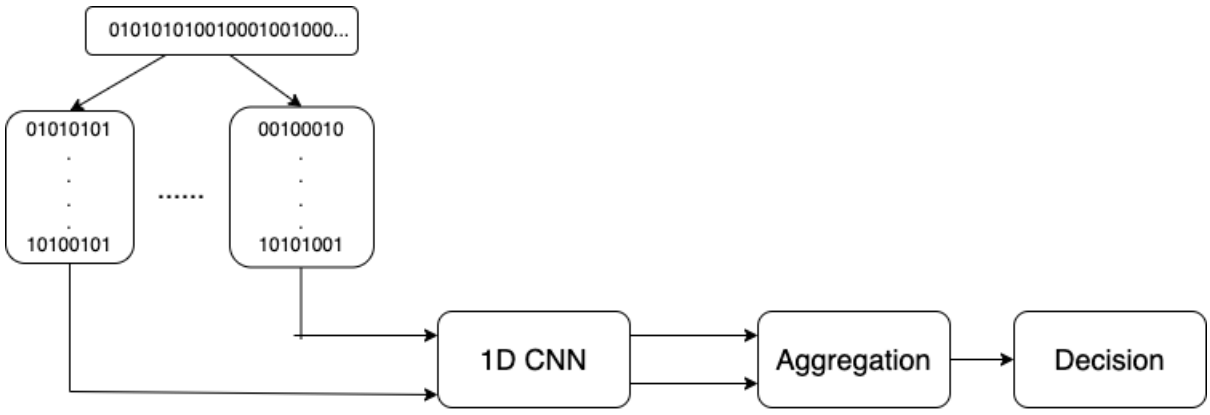
1.3 Experiments and Results

- The accuracy of the classifier tells us how confidently it is able to differentiate between multiple malware classes.
- Accuracy of the 1D CNN trained on a 90:10 train:test split was around 79%

```
(base) arpank@gpu:/data/pkalwar/malwareproject$ LD_PRELOAD=./library.so /bin/ls -l
prediction probabilities: [0.49766386 0.14323486 0.00659995 0.00305915 0.00488492 0.116209
 0.00282881 0.16119444 0.06432495]
predicted class=Ramnit with probability=0.49766385555267334
suggestion: continue with execution. Proceed? [y/n]
y
total 28
drwxrwxrwx 3 pkalwar ms 4096 Apr 29 00:41 data
-rw-r--r-- 1 pkalwar ms 2124 Apr 28 23:54 library.c
-rwxr-xr-x 1 pkalwar ms 8336 Apr 28 23:55 library.so
-rwxr-xr-x 1 pkalwar ms 4541 Apr 29 00:41 predict.py
(base) arpank@gpu:/data/pkalwar/malwareproject$ LD_PRELOAD=./library.so /bin/ls -l
predicted class=Ramnit with probability=0.49766386
file present in whitelist. executing
total 28
drwxrwxrwx 3 pkalwar ms 4096 Apr 29 00:41 data
-rw-r--r-- 1 pkalwar ms 2124 Apr 28 23:54 library.c
-rwxr-xr-x 1 pkalwar ms 8336 Apr 28 23:55 library.so
-rwxr-xr-x 1 pkalwar ms 4541 Apr 29 00:41 predict.py
(base) arpank@gpu:/data/pkalwar/malwareproject$
```

2 Architecture

2.1 Project Architecture

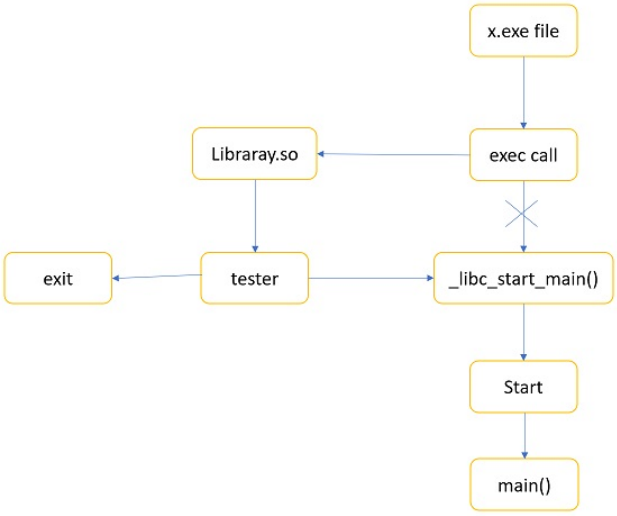


- Binary files are broken up into equi-sized chunks (16384) to be provided as input to the 1D CNN
- Results from chunks are aggregated using sum rule and the decision is made

2.2 1D CNN Architecture

- 6 Convolutional layers + 2 Fully Connected layers are used with 612,105 trainable parameters.
- MaxPooling is used between convolution layers for reducing data dimension.
- Leaky ReLU activation is used in hidden layers and Softmax activation in the last layer.
- Training is done for 100 epochs with Early Stopping.

2.3 Linux Automation Architecture



- Any function enters main from _libc_start_main built inside glibc in linux found in libc.so.6

- The custom library overrides this builtin function used by loader when loading any executable
- This allows me to intercept any executable before starting and pass to the main tester
- The export of LD_PRELOAD allows us to make it global and across all users

3 Conclusion

- We have successfully implemented 2 interactive applications using 1D CNN to detect whether an executable file belongs to some malware class with predicted probability.
- Linux application does this detection even before the start of execution of the executable and thus, doesn't execute the file if found malicious.
- Porting the linux automation to Windows would have immense applications.

4 Limitations

- Significant delay on testing for a fresh binary file.
- Accuracy of ML model can be improved using better compression and resizing techniques

5 References

- Efficient Malware Classification by Binary Sequences with One-Dimensional Convolutional Neural Networks
- Malware Classification with Deep Convolutional Neural Networks
- Malware Classification Using Image Representation