Automated Malware Classification using Deep Learning

Group Name: AccessDenied

Members:

- Arpan Kapoor
- Chabil Kansal
- Kaustubh Verma
- Manish Kumar
- Vikas Lodhi
- Punyabrat Kalwar

Roll No.:
21111016
21111022
190424
21111037
21111068
21211403

Supervised By : Prof. Sandeep Kumar Shukla

BRIEF OVERVIEW:

- A Malware Detection Application using Deep Neural Networks to classify binary executable files
- The application scans 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.
- The Application uses a 1-D CNN model for Classification.

- In linux, for getting the executable file about to run we modified the libc environment initialisation function to send the file for scanning before execution.
- In Windows, we have developed a GUI application which can take the path of an executable file and give us the prediction whether the file belongs to a malware class or is benign in nature
- We have also implemented a white list which maintains the file path, its hash value along with it's predicted probability and malware class.
- This whitelist helps to speed up the subsequent execution of an already scanned executable file.

Literature Review

Efficient Malware Classification by Binary Sequences with One-Dimensional Convolutional Neural Networks

- This paper proposed a byte-level 1D CNN model to explore informative features from the one-dimensional structure of binary executables. The experimental results in this paper shows that their 1D CNN model achieves promising results by giving smaller resizing bit/byte-level sequences.
- 1D CNNs achieved better or comparable results with less computational cost compared with 2D CNNs in terms of the amount of multiply-add operations.

TOPICS
Dataset & Preprocessing
1-D CNN and It’s Implementation
Linux Malware Autodetection
System Application Implementation
Advantages & Limitations
Demo

Dataset

- Microsoft Malware Classification Challenge (BIG 15) dataset
- https://www.kaggle.com/c/malware-classification/data
- 10868 samples of malware belonging to 9 classes

Label	Malware Family	Size
0	Ramnit	1541
1	Lollipop	2478
2	Kelihos_ver3	2942
3	Vundo	475
4	Simda	42
5	Tracur	751
6	Kelihos_ver1	398
7	Obfuscator. ACY	1228
8	Gatak	1013

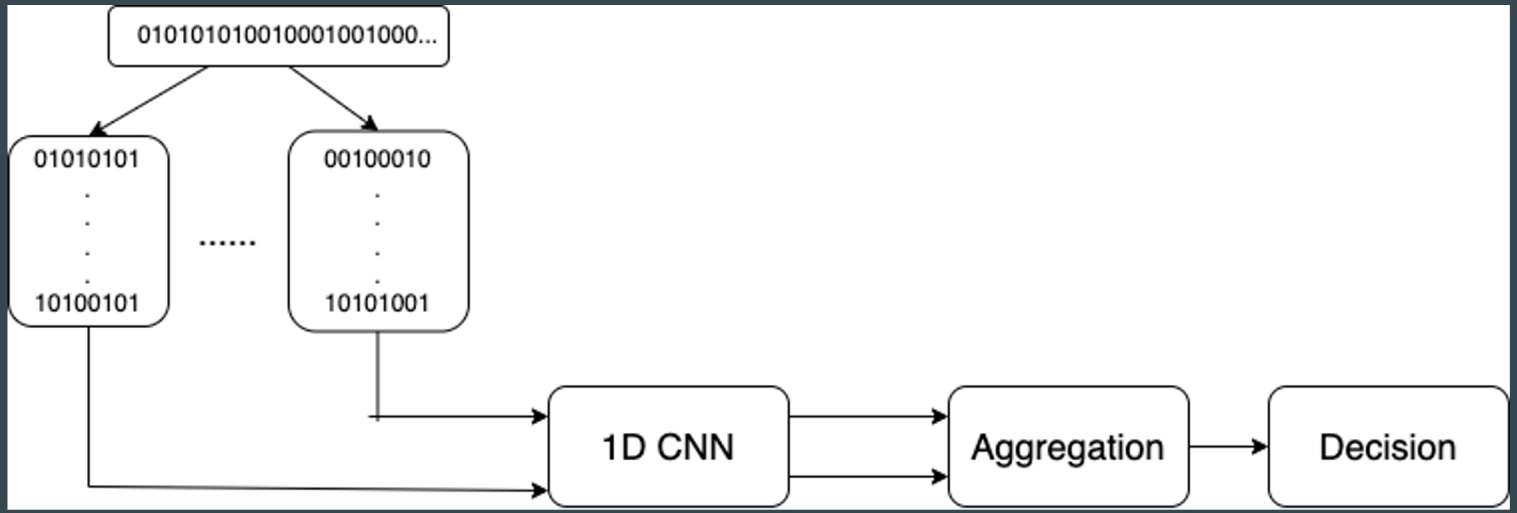
Preprocessing

- `.bytes` files in hex dump form (without PE header) are converted to binary form

```
00401000 C7 01 54 03 41 00 E9 38 26 00 00 CC CC CC CC CC
00401010 56 8B F1 C7 06 54 03 41 00 E8 25 26 00 00 F6 44
00401020 24 08 01 74 09 56 E8 AD 26 00 00 83 C4 04 8B C6
00401030 5E C2 04 00 CC CC CC CC CC CC CC CC CC CC CC CC
00401040 33 C0 C2 10 00 CC CC CC CC CC CC CC CC CC CC CC
00401050 8B C1 8B 88 A0 00 00 00 53 56 8B 58 38 57 8B 78
00401060 64 8B D7 D3 E2 66 8B 48 78 FF 80 BC 00 00 00 66
00401070 83 C9 1A 89 50 20 8D 90 A4 00 00 00 66 89 48 6C
00401080 0F B7 0A 8B F1 83 CE 64 66 89 70 48 0F B7 B0 C0
00401090 00 00 00 66 0B F1 66 89 B0 B4 00 00 00 0F B7 70
```

- Binary files are broken up into equi-sized chunks (16384) to be provided as input to the 1D CNN

- Last chunk is padded with zeros if needed
- Results from chunks are aggregated using sum rule



1D CNN Motivation

Our motivation behind using 1D CNN is twofold:

1. In our survey, we found that 1D CNNs can give comparable results with less computational cost compared with 2D CNNs in essence contributing user friendliness.
2. The key difference between 1D CNN and 2D CNN models is that we explore the informative features by following the original one-dimensional structure of binary executables. 2-dimensional conversion could twist the sequential structure of the machine codes since a fixed width will cut the sequential binary codes. More specifically, the binary codes representing a certain behavior might be broken into pieces because the image's width must be determined. Rather than convert the malware executables into images, we aim to apply one-dimensional CNNs to the binary codes for maintaining the sequential structures

1D CNN architecture

- 6 CONV layers + 2 FC 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 in the last layer
- 90% of the dataset is used for training and 10% as test
- Training is done for 100 epochs with Early Stopping

Linux Malware Autodetection

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

Advantages & Limitations

- Application does not interfere with any IDS system like wazuh or snort
- It does not waste CPU cycles on running a benign query again or when system is idle
- It only starts up when an executable is about to run
- Unlike a NIDS it doesn't scan entire packet stream to check for malware , as it directly checks the exe file.
- Specially useful in data centers where the typical antivirus technique leads to a lot of wastage as all data are scanned as opposed to only the executables that are run.
- Having a ML based classifier helps to avoid the issues faced by the signature based antivirus who cant detect auto modifying malwares and new malwares
- It is quite user friendly and easy to install compared to Wazuh

Con

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

Linux and Windows application demo

Linux Demo link
Windows demo link

Conclusion

- We 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.

Thank You!