Dynamic Malware Prediction

CS658A PROJECT REPORT

Team - Turing

Ashutosh Patel	21111018	ashutoshp21@iitk.ac.in
Deepak Raj	21111024	deepakr21@iitk.ac.in
Dinkar Tewari	21111025	dinkart21@iitk.ac.in
Harsh Agarwal	21111030	harshmag21@iitk.ac.in
Rohit Kushwah	21111053	krohit21@iitk.ac.in
Varun Vankudre	21111064	varunsv21@iitk.ac.in

Indian Institute of Technology, Kanpur

Keywords : Malware Detection, Cuckoo Sandbox, Dynamic Analysis, Recurrent Neural Networks

Abstract

Most of the old anti-virus system uses static malware detection techniques. These systems quickly examine the features of an executable code and match to previously observed code. However, they are vulnerable to code obfuscation techniques. It takes a long time to catch behavioral data of such obfuscated codes - almost 5 minutes, by the time malware payload has likely to be already delivered even before detection. In this project, we tried to build a system that will classify an executable as malicious or benign by looking at a short snapshot of behavioral data. We are able to predict an executable by just looking at the first 10 seconds of the execution using recurrent neural networks. This way, we can even predict malware that has no previous record.

1 Introduction

Malware is an acronym for malicious software, which is designed to harm computer without the user's awareness. There are various kind of malware's such as Trojans, rat's, ransomware, worms and rootkits. Malware is key of various vulnerabilities. Many companies struggle to comprehend the malware that they come across. Understanding how to detect the malwares by programming means can solve malware detection issue worldwide. Talking about online platform VirusTotal, a free platform which can be used to evaluate whether files are malicious, regularly approaches one million new, distinct files for analysis each day. Such platforms are not capable of predicting early stage malwares hence, doesn't meet malware checking requirments as we thought it would.

There are certain malware analysis methods at present such as static and dynamic analysis. Static Analysis is the automated analysis of source code without executing the application. It is often used to detect security vulnerabilities, performance issues etc. whereas Dynamic analysis is the process of testing and evaluating a program — while software is running. Also referred to as dynamic code scanning, dynamic analysis improves the diagnosis and correction of bugs, memory issues, and crashes of an application during its execution.

The main contributions of this report are

1. We propose a recurrent neural network (RNN) model to predict malicious behaviour using machine activity data and demonstrate its capabilities are superior to other machine learning solutions that have previously been used for malware detection

2. We conduct a case study using 1498 malicious samples and show that our model has high detection accuracy at some second into execution without prior exposure to malwares.

2 Dataset

2.1 Data Collection

We needed both malware and benign to train our model. We have taken almost 1500 malware from the mentioned link "malwarebazar.com." We downloaded different applications from accessible sources such as Softonic, Filehippo, and SourceForge for benign. We classified these downloaded applications using VirusTotal. We also used Windows 7 executables as benign applications. Roughly, we used around 1000 benign files.

2.2 Input Features

The total of ten features collected are Total processes, max process id, CPU user(%), CPU system(%), Memory use(Bytes), Swap use(Bytes), packets sent, packets Received, Bytes Sent, Bytes received.

Feature	Min Value	Max Value
Total Process	46	98
Max Process ID	3428	11728
System CPU Usage	0%	94.3%
User CPU Usage	0%	28.1%
Memory Use	528441344	1484062720
Swap Memory Use	0	1005510656
Transferred Packets	1123	510255
Received Packets	2785	707099
Transferred Bytes	181596	981187893
Received Bytes	3823939	1026630330

Figure 2: Feature Set Min and Max Values

3 Methods

3.1 Cuckoo Setup

After collecting the complete dataset, the next step is to collect features from the dataset to feed it to our RNN model and make the predictions.
We set up the cuckoo sandbox in the windows-7 host for collecting the dataset. The complete setup is done on VirtualBox. To obtain the features from the sandbox, we need to submit the file to the sandbox. The sandbox will run the malware file in a safe environment and return the host machine's features. The virutal machine was restarted between each sample execution to ensure that malicious and benign files alike began from the same machine set-up.

We have written our custom python script using the psutil library to collect machine activity data. The script will run in the cuckoo sandbox (a virtualized sandboxing tool) along with the malware and gather per second machine activity data. In order to run this python script, psutil must be installed in the cuckoo sandbox. We enabled the internet in the cuckoo sandbox and installed the psutil library.

We use ten machine activity data metrics as feature inputs to the model. We take a snapshot of the metrics every second for 30 seconds while the sample executes, starting at 0s, such that at 1s, we have two feature sets or a sequence length of 2. The metrics captured were: system CPU usage, user CPU use, packets sent, packets received, bytes sent, bytes received, memory use, swap use, the total number of processes currently running, and the maximum process ID assigned. A key advantage of continuous data such as machine activity metrics is that the model can infer information from completely unseen input values. Any unseen data values in the test set will still have numerical relevance to the data from the training set as they will have a relative value that can be mapped onto the learned model. The machine activity data we collected are continuous numeric values, allowing for a large number of different machine states to be represented in a small vector of size 10.

As the data is sequential, we chose an algorithm capable of analyzing sequential data. Using the time-series data means that the rate and direction of change in features and the raw values themselves are all inputs to the model. Recurrent Neural Networks (RNNs) and Hidden Markov Models can capture sequential changes. However, RNNs hold the advantage in situations with a large possible universe of states and memory over an extended chain of events. They are therefore better suited to detecting malware using machine activity data.
We anticipate that the RNN should be re-trained regularly with newly discovered samples, thus the architecture may need to change too. As it needs to be carried out multiple times, this process should be automated.

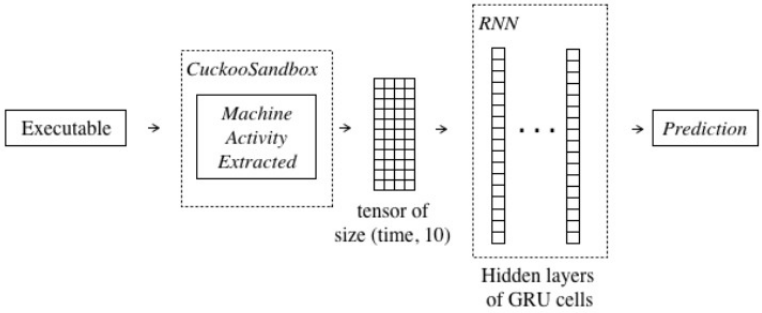
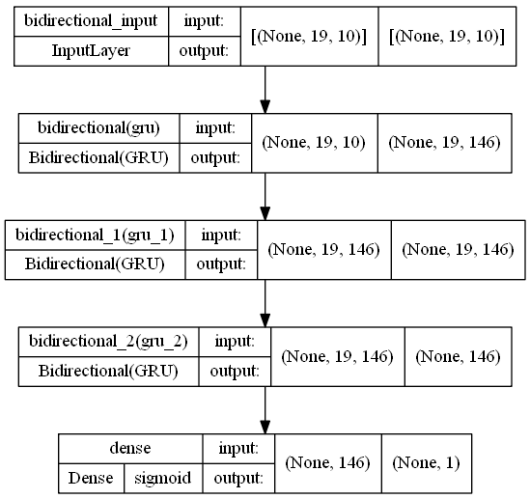


Figure 3: Process Flow

3.2 Training RNN

The feature set generated is such that it is sequential. Due to this property of the training dataset, we chose to use Recurrent Neural Network as it has been proved to be able to use time-series data. The rate and direction of change in feature value and raw feature values are all input to the model. RNN also performs well with a sizeable possible universe of states and memory over a long chain of events. An RNN creates a temporal depth when multiple hidden layers are used similarly to other artificial neural networks. We have used Gated Recurrent Units (GRU) instead of Long-Short Term Memory (LSTM) as previous research has shown a shorter training time in most cases. Bidirectional RNNs use two layers in every hidden layer, one processing the time series progressively, and the second processing regressively



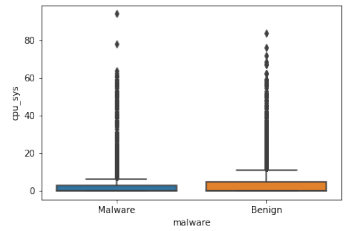
(a) RNN ARCHITECTURE

Hyper Parameter	Optimal Value
Cell Type	GRU
Loss	Binary Cross-Entropy
Activation Function	Sigmoid
Optimiser	Adam
Sequence Length	10
Recurrent Dropout	0.1
Depth	3
Bi-Directional	True
Hidden Neurons	73
Epoch	70
Batch Size	64

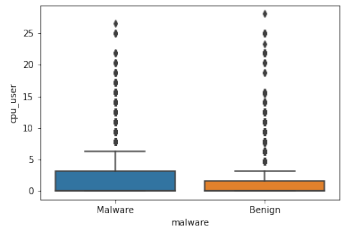
(b) RNN HYPER PARAMETERS

4 Results and Observations

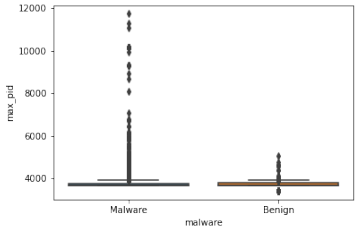
We have divided our data into test and training sets. On sequence length of 10 seconds, our accuracy is around 92%. We also observed that accuracy increases on increasing the sequence length, and on 19 seconds, we are getting a maximum accuracy of around 97%. However, we stick on to 10 seconds as we believe running malware for a longer time will put a user's system at risk of vulnerability. The figures below show the input features' frequency distributions for benign and malicious samples.



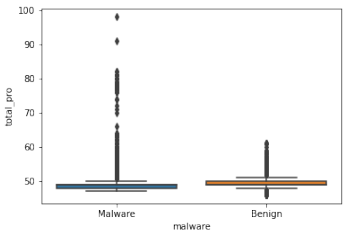
(c) System CPU Usage



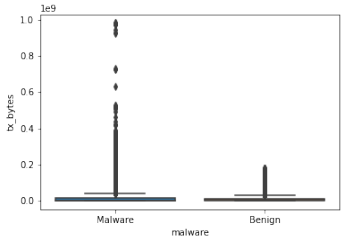
(d) User System Usage



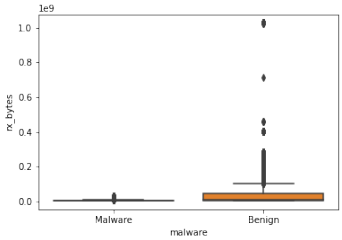
(e) Max Process ID



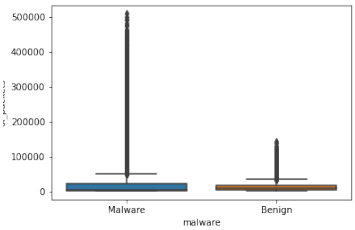
(f) Total No of Process



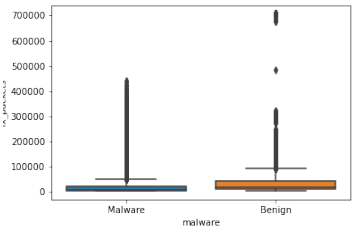
(g) Transferred Bytes



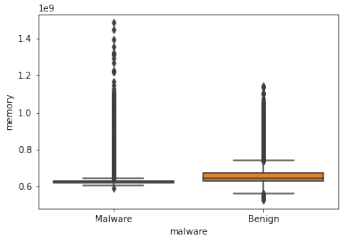
(h) Received Bytes



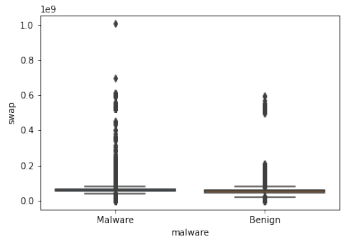
(i) Transferred Packets



(j) Received Packets



(k) Memory Usage



(l) Swap Memory Usage

Figure 4: Frequency distribution of input features for benign and malicious files

5 Limitations and Future Work

- So far, we have built our model only for Windows 7 executable. This model can be extended to predict the malicious activities of PDFs, URLs and other malware, and applications.
- The model can also classify unseen malware significantly if the training data is diverse enough. So, we need to collect data over a long period to get different kinds of malware.
- Currently, we are extracting features from the cuckoo sandbox only. We can extend this project to make an application that will take real-time features of a running application and block an application if malware is detected, just like any anti-virus.

6 Conclusion

Dynamic malware detection methods are often preferred to static detection as the latter are particularly susceptible to obfuscation and evasion when attackers manipulate the code of an executable file. However, dynamic methods previously incurred a time penalty due to the need to execute the file and collect its activity footprint before making a decision on its malicious status. This meant the malicious payload had likely already been executed before the attack was detected. We have developed a novel malware prediction model based on recurrent neural networks (RNNs) that significantly reduces dynamic detection time, to less than 10 seconds per file, whilst retaining the advantages of a dynamic model. This offers the new ability to develop methods that can predict and block malicious files before they execute their payload completely, preventing attacks rather than having to remedy them.

References

[Baz] Malware Bazar. Malware Download.

[For] Source Forge. Portable Executables Download.

[Hip] File Hippo. Portable Executables Download.

[JNR+18] Sainadh Jamalpur, Yamini Sai Navya, Perla Raja, Gampala Tagore, and G Rama Koteswara Rao. Dynamic malware analysis using cuckoo sandbox. In *2018 Second international conference on inventive communication and computational technologies (ICICCT)*, pages 1056–1060. IEEE, 2018.

[LBE15] Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

[OM13] Digit Oktavianto and Iqbal Muhardianto. *Cuckoo malware analysis*. Packt Publishing Ltd, 2013.

[RBJ18] Matilda Rhode, Pete Burnap, and Kevin Jones. Early-stage malware prediction using recurrent neural networks. *computers & security*, 77:578–594, 2018.

[Tot] Virus Total. Virus Total Malaware Check.