Malware Detection by Exploiting Deep Learning over Binary Programs

Panpan Qi*, Zhaoqi Zhang*, Wei Wang* and Chang Yao†

*School of Computing, National University of Singapore, Singapore
†Institute of Computing Innovation, Zhejiang University, China

Introduction

Motivation

- Malware (malicious software) remains the most popular and damaging attack vector, costing hundreds of billions in damage.
- Malware evolves rapidly, with reports showing that 99% disappear after 58 seconds.
- Traditional machine learning models heavily depend on feature engineering and could be easily deceived by hackers.
- Anti-virus industry prefer to increase the recall (i.e., true positive rate) while maintaining a low false positive rate (usually less than 0.1%).

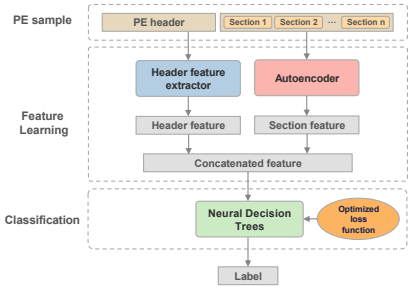
Contribution

- Proposed an end-to-end malware detection framework based on deep learning techniques, which achieves the best performance among existing deep learning based methods.
- Proposed an effective loss function for optimizing recall with a fixed tiny false positive rate.
- Conducted experiments on a real large dataset to confirm the effectiveness of the proposed feature learning framework and loss function for malware detection.

Approach

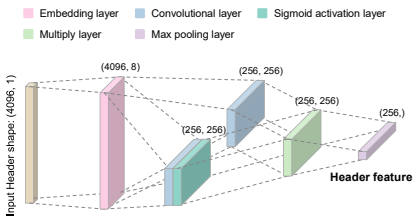
Model Architecture

- **Feature Learning**
  - Two separate structures for processing PE header and PE sections
- **Classification**
  - Neural Decision Trees and logistic regression
  - A special loss function for optimizing the recall given a fixed false positive rate



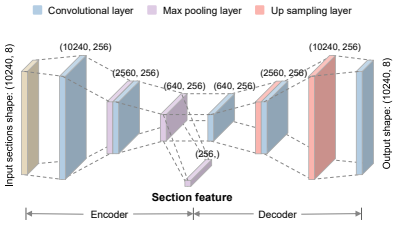
Header Feature Extraction

- **Input**
  - Raw byte sequence of the PE header
- **Embedding layer**
  - Embeds the raw bytes into a continuous and distributed representation
- **Gated Convolution layer**
  - $X_A \otimes \sigma(X_B)$
  - Provides a mechanism to learn, select and pass along the important and relevant information
- **Global Max-pooling layer**
  - Produces the header feature regardless of the location of the detected features



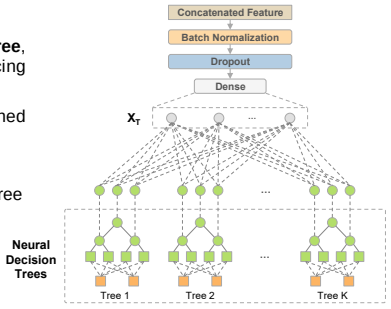
Section Compression

- **Input**
  - Executable sections in PE sample
- **Encoder**
  - Conv + Max pooling layer
- **Decoder**
  - Conv + Up sampling layer
- **Loss function**
  - Mean square error



Neural Decision Trees

- A **differentiable version of decision tree**, enabling end-to-end training and reducing overfitting.
- Two ensemble techniques are designed for Neural Decision Trees:
  - Neural Random Forest
  - Neural Gradient Boosting Decision Tree
- Logistic regression is applied on all the outputs of the decision trees, providing a more flexible way of utilizing the generated trees.



Loss Function Optimization

- To maximize recall rate with the restriction that false positive rate ≤ 0.1%.

$$L = \frac{\sum_{i \in Y^+} l(x_i, y_i)}{|Y^+|} + max\left(0, \lambda\left(\frac{\sum_{i \in Y^-} l(x_i, y_i)}{|Y^-|} - \alpha\right)\right)$$

Loss for positive samples

Loss for negative samples

Experiments

Data Summary

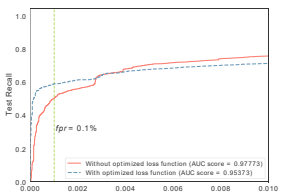
- SecureAge deployed 12 commercial antivirus engines that are continuously scanning data from the endpoints.
  - Positive: num of engines >= 4
  - Negative: num of engines = 0

Dataset	Positive samples	Negative samples
February	110656	80185
March	100651	92097
April	58394	48595
May	42635	87858

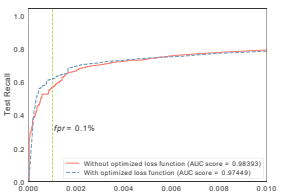
Experiment Results

Training Dataset	Test Dataset	Model	Without optimized loss function		With optimized loss function	
			AUC (%)	Recall (%)	AUC (%)	Recall (%)
February	March	MalConv	95.45±0.34	33.58±16.21	94.79±0.32	53.17±4.37
		ConvNet	96.21±0.17	45.11±3.88	94.34±0.60	49.92±3.69
		EntropyNet	91.61±0.22	33.88±9.13	88.13±0.73	41.52±4.38
		Proposed Model	**96.47±0.20**	**56.14±3.65**	**96.40±0.19**	**57.52±2.95**
March	April	MalConv	98.50±0.12	50.67±11.75	98.21±0.31	57.41±9.74
		ConvNet	98.82±0.12	63.67±5.50	98.27±0.70	67.39±5.69
		EntropyNet	95.70±0.32	24.53±6.76	93.95±0.48	49.68±8.09
		Proposed Model	**99.16±0.04**	**71.54±3.32**	**99.12±0.07**	**75.25±1.62**
April	May	MalConv	97.95±0.36	52.28±8.12	94.02±1.48	58.55±2.43
		ConvNet	98.33±0.26	55.91±2.68	96.66±0.73	56.96±3.45
		EntropyNet	90.96±0.96	31.33±3.13	81.94±2.33	35.23±3.24
		Proposed Model	**98.60±0.20**	**70.29±1.03**	**98.43±0.35**	**70.69±0.93**

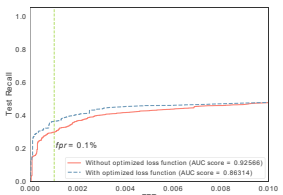
ROC Curves Comparison



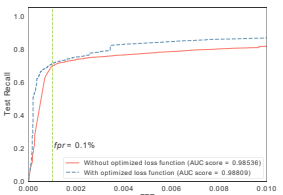
(a) ROC curves of MalConv



(b) ROC curves of ConvNet



(c) ROC curves of EntropyNet



(d) ROC curves of Proposed Model