Machine-Learning Based Malware Classification

Miguel Oyler-Castrillo[1], Nicolas Agostini[2], Gadiel Sznaier[2], David Kaeli[2],
1.migueloyler@gmail.com, bohmagostini.n@husky.neu.edu, sznaiercamps.g@husky.neu.edu, kaeli@ece.neu.edu
Massachusetts Bay Community College, Northeastern University

NSF Award Number 1559894

Abstract

Modern day anti-virus software can quickly detect *known* malicious programs that exist in databases containing hundreds of millions of malware signatures. Today's malware can alter its signature to avoid detection. In this project, we seek to develop a ***machine-learning*** model that accurately detects previously unseen malware based on the program's x86 instruction sequence

Noise Reduction Strategy

- To reduce the amount of noise in our data, we compute the `term-frequency (tf) * inverse-document-frequency (idf)` score for each instruction in each file in our dataset
- We then use the `tf*idf` score as input to a Random Forest Classifier (RFC)
- Using the output of the RFC, we can find and remove non-informative instructions from the instruction sequences [1], before passing them as input to a Recurrent Neural Network [2]

Feature Selection and Preprocessing

Parse	*Extract sequences of opcodes from x86 assembly files*
Count	*Use frequency statistics to quantify rarity of opcode with respect to the entire dataset*
Filter	*Use opcode `tf*idf` scores as input to a Random Forest Classifier (RFC)*
Reduce	*Parse through opcode sequences, keeping only the most informative opcodes based on the RFC output*
Vectorize	*Convert opcode sequences into vector sequences, to ease use as input for a Recurrent Neural Network*
Train Test	*Fit and train the Recurrent Neural Network*

Opcode Extraction, Noise Reduction, and Vectorization

```
long arith
(long x, long y, long z)
{
  long t1 = x+y;
  long t2 = z+t1;
  long t3 = x+4;
  long t4 = y * 48;
  long t5 = t3 + t4;
  long rval = t2 * t5;
  return rval;
}
```

Original C / C++ code

`(x,y,z) -> (%rdi,%rsi,%rdx)`

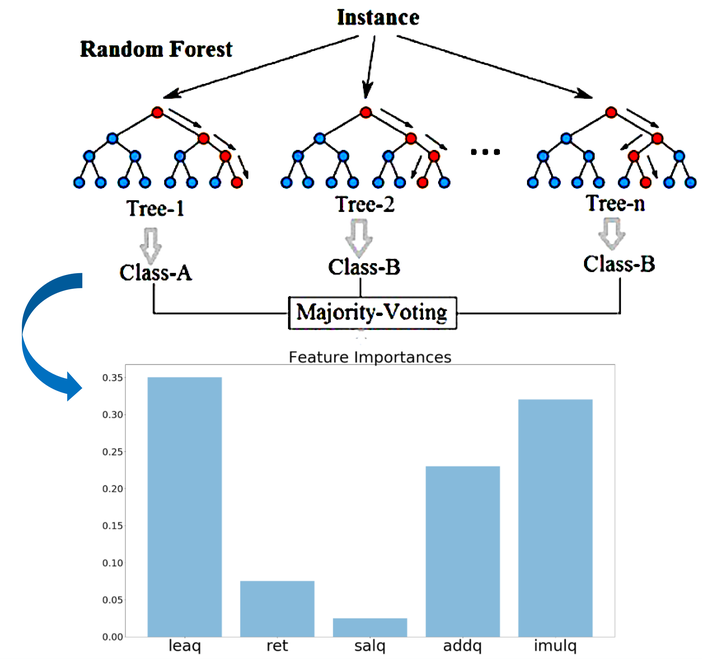
```
arith:
  leaq   (%rdi,%rsi), %rax
  addq   %rdx, %rax
  leaq   (%rsi,%rsi,2), %rdx
  salq   $4, %rdx
  leaq   4(%rdi,%rdx), %rcx
  imulq  %rcx, %rax
  ret
```

Intel x86 Assembly Instructions

['leaq', 'addq', 'leaq', 'salq', 'leaq, 'imulq', 'ret']

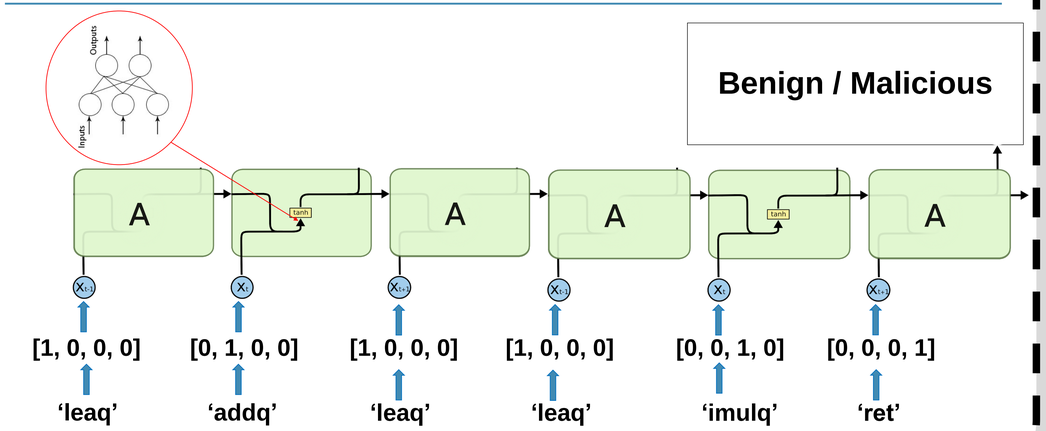
TF * IDF

Opcode	leaq	addq	salq	imulq	ret
TF * IDF score	0.43	0.14	0.14	0.14	0.14

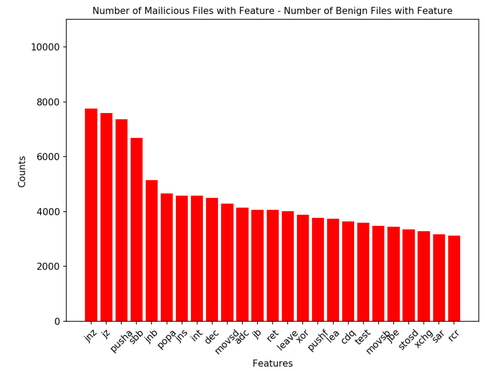


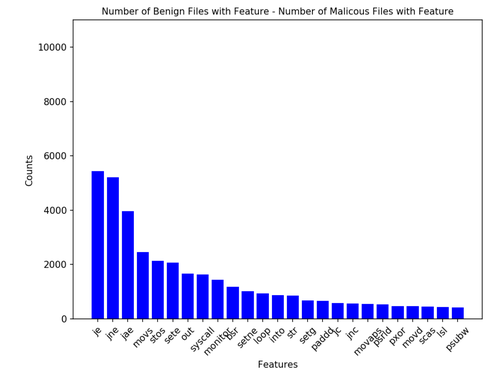
['leaq', 'addq', 'leaq', 'salq', 'leaq, 'imulq', 'ret']

['leaq', 'addq', 'leaq', ~~'salq'~~, 'leaq', 'imulq', 'ret']

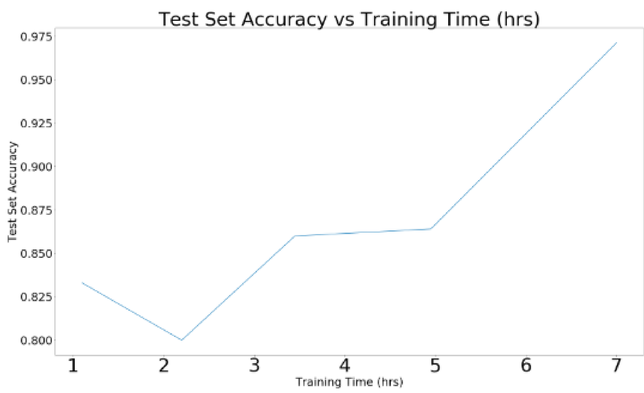


Benign / Malicious





Results



Acknowledgements

I would like to thank my advisor, Dr. Kaeli, as well as my mentors Nicolas Bohm Agostini and Gadiel Sznaier, for the outstanding support they've given me over the course of this program.

References

[1] Jinpei Yan, Yong Qi, and Qifan Rao, "Detecting Malware with an Ensemble Method Based on Deep Neural Network," Security and Communication Networks, vol. 2018, Article ID 7247095, 16 pages, 2018

[2] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997