Malware Classification using File Content and Characteristics

EN.650.672.01 Security Analytics

By Kandarp Khandwala & Antara Sargam

Introduction

Motivation

- Malware Industry
  - Well organized market
  - Well-funded. Large amount of money involved
- Traditional Protection
  - Constantly evolving threats
  - Polymorphism introduced in malware
  - Difficult to detect by anti-malware software

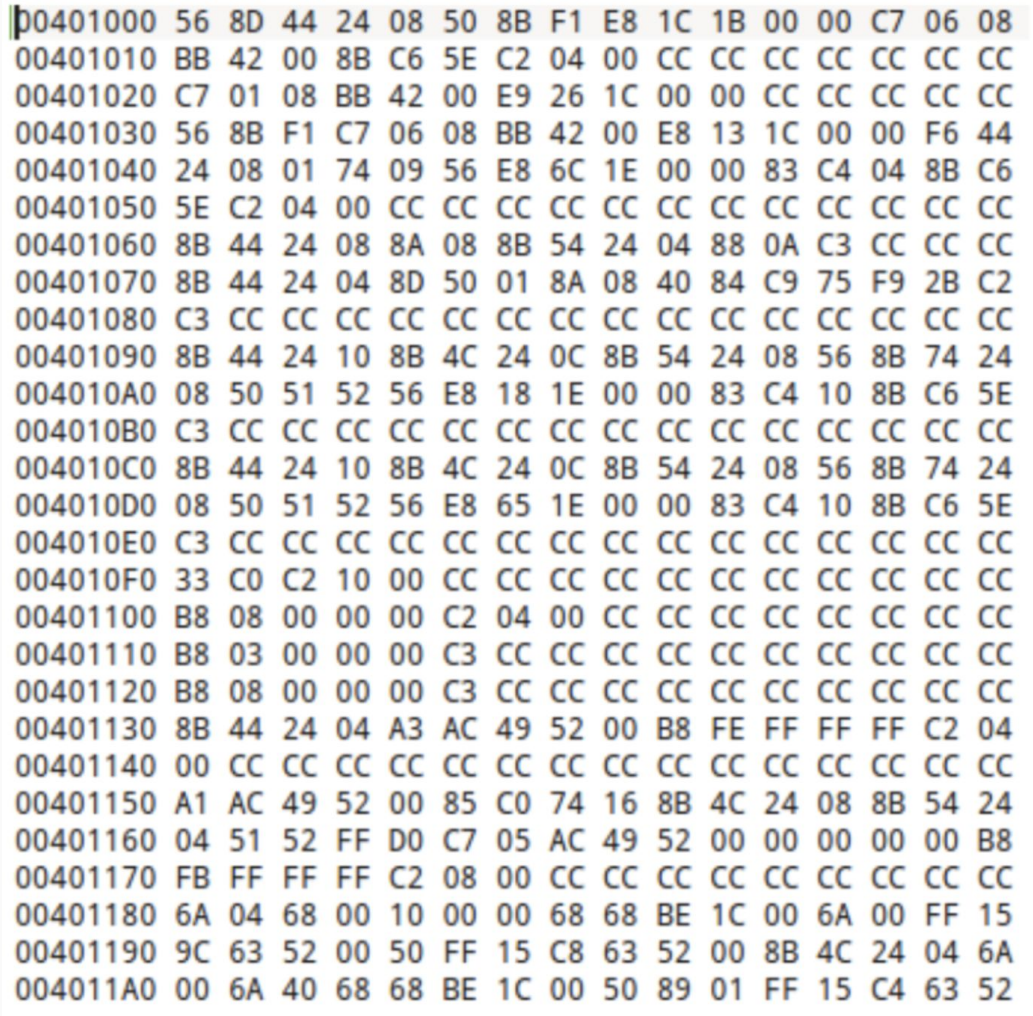
Understanding the Dataset

Microsoft malware dataset

- Consists of a separate training and test data set.
- Both training and test dataset are a combined size of around half a terabyte!
- Around 10,000 malware files for every dataset. (we did not use the entire set due to challenges ~ 2000 files)
- Each malware provided in 2 versions of the file: **bytes file** and **asm file.**
- Each malware file has an ID, a 20-character hash value uniquely identifying the file, and a Class, an integer representing one of 9 family names to which the malware may belong.

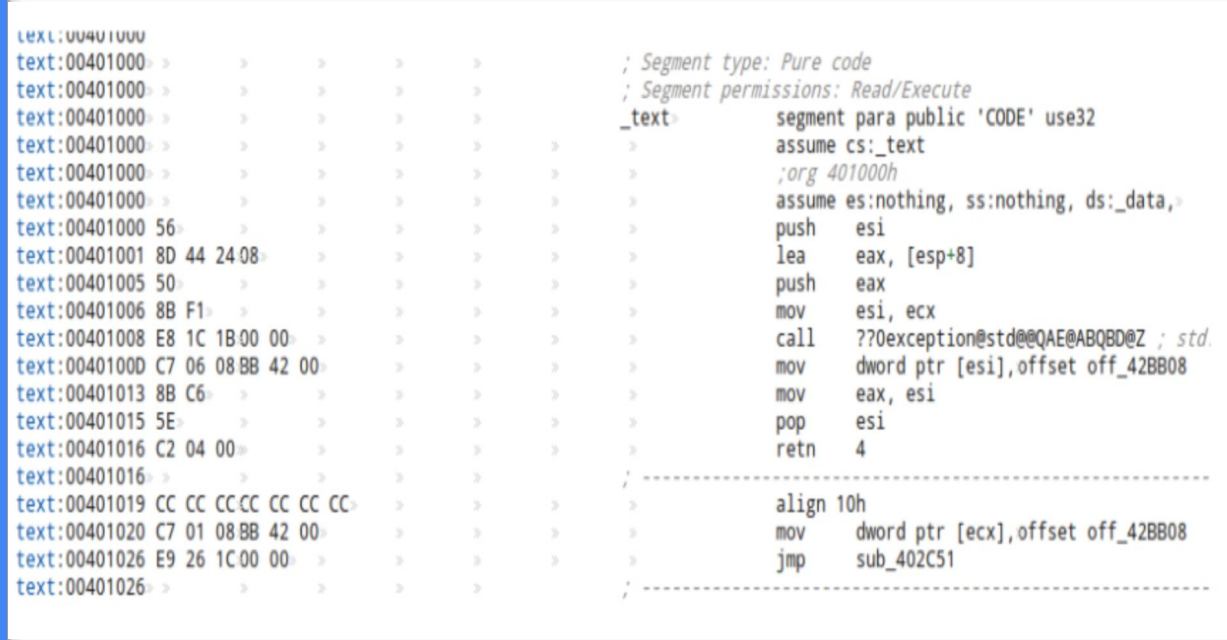
Bytes File

- A simple binary representation of the malware file
- Features Extracted:
- File size
- Compression ratio
- Entropy of the file, etc.



ASM File

- Contents of the file were extracted by running the IDA Pro disassembler on the binary file.
- Features used:
- File size
- Compression ratio
- Contents over 2-gram
- String file characteristics like url, registries, headers, etc.



Target Value: Malware “class”

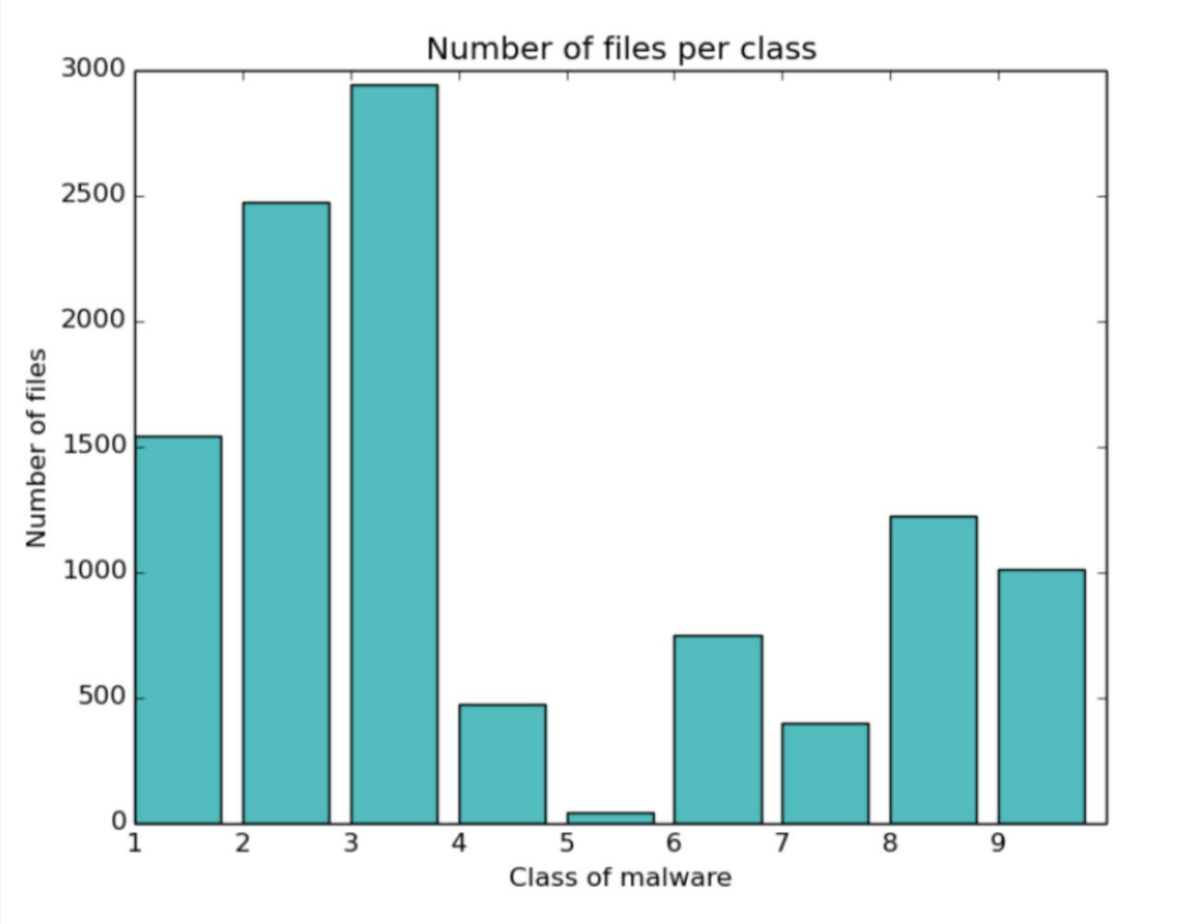
We were given the challenge to predict the class or “family” the test malware files belonged to with the highest accuracy.

Malware Class

- Each malware file in the dataset belonged to one of the 9 malware families.
- Each malware family was represented by a decimal number between 0-9
- Our target was to predict the class the new set of malware family belonged to using our model.

- Ramnit
- Lollipop
- Kelihos_ver3
- Vundo
- Simda
- Tracur
- Kelihos_ver1
- Obfuscator.ACY
- Gatak

Number of malware files per class



Model Selection

Classification Models

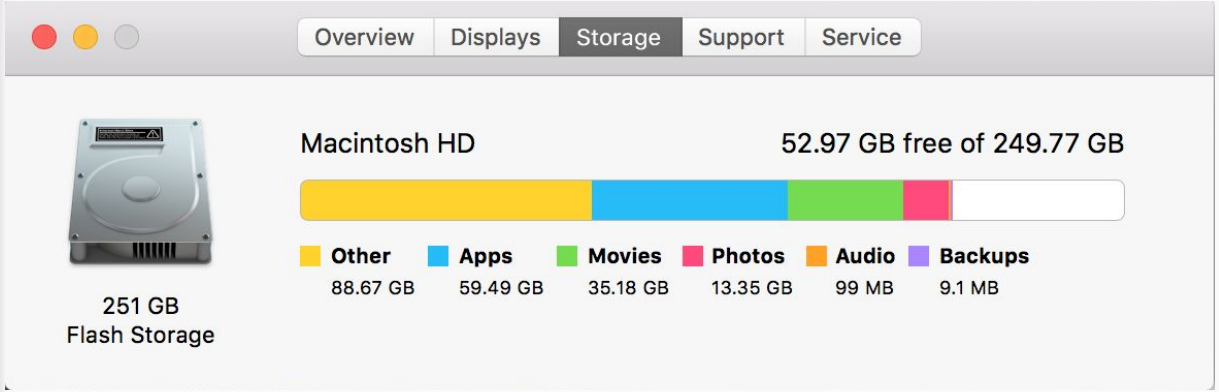
We tried to create the best model which gives us the highest accuracy on the test dataset. We tried the following classification models:

- Extra Trees Classifier
- LightGBM

Model training & techniques

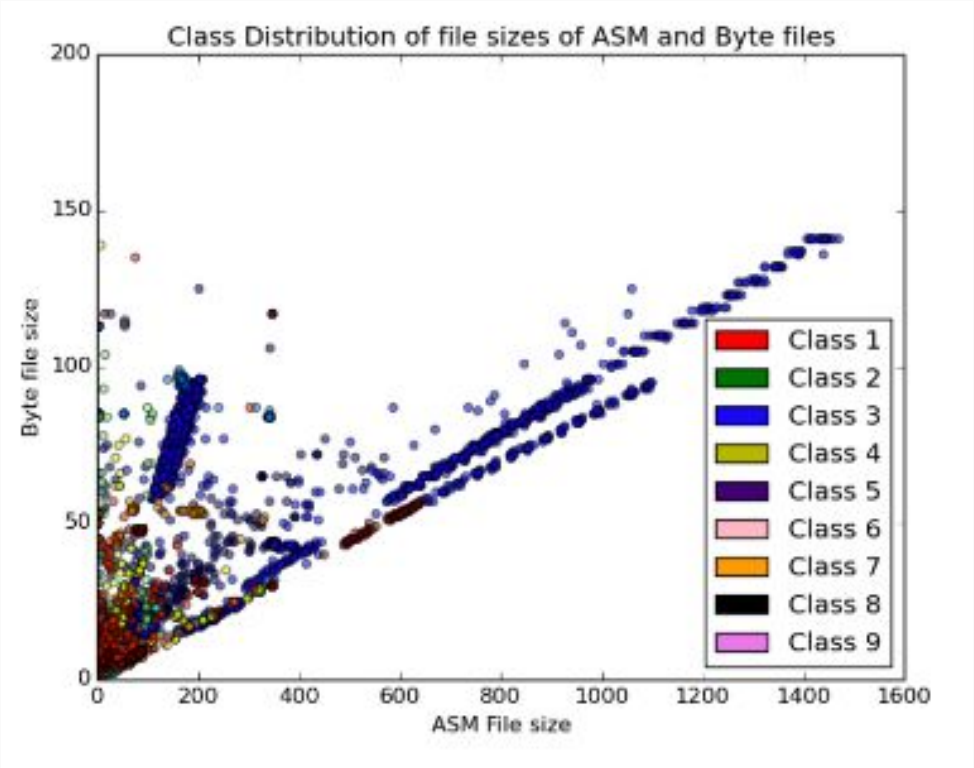
Dealing with Large Dataset

- Out of storage in a blink of an eye
- AWS EC2 and S3 calls slow
- ~2 hours just to read the file
- RAM hungry
- Out of memory errors
- Reduced to ~2000 files of about 150GB
- Can be improved with models with batch processing



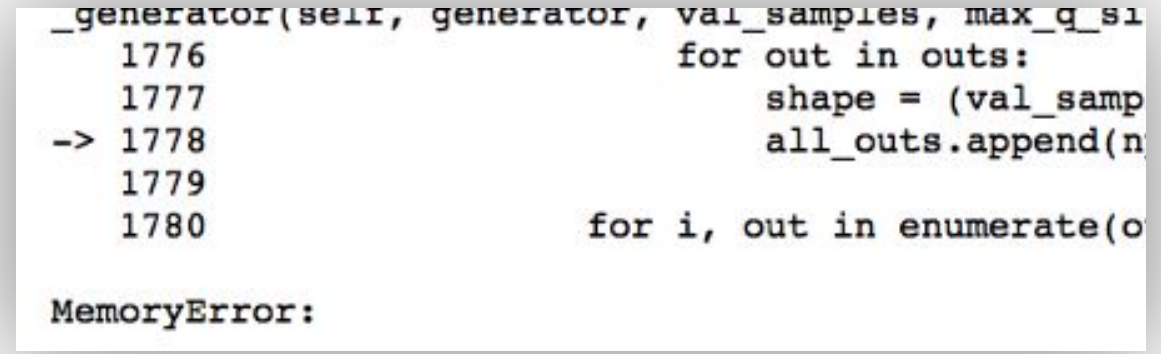
File Size

- Scatter Plot showing .asm and .bytes file sizes with colors representing malware families
- Compressed file size also another picture
- Clear distinction showing class segregation



Vectorizer

- TFIDFVectorizer
  - loads the input matrix into memory
  - expensive process
- HashingVectorizer
  - very low memory
  - scalable to large datasets
  - no need for vocabulary dictionary in memory
  - 85% accuracy
- Streaming/Batch Processing in the future



String Characteristics

- Occurrences of
  - the string 'C:\'.
  - http:// or https://
  - Registries HKEY_
  - DOS MZ executable
- Entropy of strings
- Other string features
- Total feature vectors now ~10108

```
1 import lief
2 # ELF
3 binary = lief.parse("/usr/bin/ls")
4 print(binary)
5
6 # PE
7 binary = lief.parse("C:\\Windows\\explorer.exe")
8 print(binary)
9
10 # Mach-O
11 binary = lief.parse("/usr/bin/ls")
12 print(binary)
13
```

Multiclass Object Detection Models

Random Forest (RF)

- Handle large amount of training data efficiently
- Inherently suited for multi-class problems

- Large number of trees make the algorithm slow

Light Gradient Boosting (LightGBM)

- Great when data is often highly unbalanced such as cyber security
- 10x faster in our case

- Harder to tune than RF
- More sensitive to overfitting if the data is noisy

Results

98%

Thanks!