

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.

00401000	56	8D	44	24	08	50	8B	F1	E8	1C	1B	00	00	C7	06	08
00401010	BB	42	00	8B	C6	5E	C2	04	00	CC	CC	CC	CC	CC	CC	CC
00401020	C7	01	08	BB	42	00	E9	26	1C	00	00	CC	CC	CC	CC	CC
00401030	56	8B	F1	C7	06	08	BB	42	00	E8	13	1C	00	00	F6	44
00401040	24	08	01	74	09	56	E8	6C	1E	00	00	83	C4	04	8B	C6
00401050	5E	C2	04	00	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC
00401060	8B	44	24	08	8A	08	8B	54	24	04	88	0A	C3	CC	CC	CC
00401070	8B	44	24	04	8D	50	01	8A	08	40	84	C9	75	F9	2B	C2
00401080	C3	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC
00401090	8B	44	24	10	8B	4C	24	0C	8B	54	24	08	56	8B	74	24
004010A0	08	50	51	52	56	E8	18	1E	00	00	83	C4	10	8B	C6	5E
004010B0	C3	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC
004010C0	8B	44	24	10	8B	4C	24	0C	8B	54	24	08	56	8B	74	24
004010D0	08	50	51	52	56	E8	65	1E	00	00	83	C4	10	8B	C6	5E
004010E0	C3	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC
004010F0	33	C0	C2	10	00	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC
00401100	B8	08	00	00	00	C2	04	00	CC	CC	CC	CC	CC	CC	CC	CC
00401110	B8	03	00	00	00	C3	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC
00401120	B8	08	00	00	00	C3	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC
00401130	8B	44	24	04	A3	AC	49	52	00	B8	FE	FF	FF	FF	C2	04
00401140	00	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC
00401150	A1	AC	49	52	00	85	C0	74	16	8B	4C	24	08	8B	54	24
00401160	04	51	52	FF	D0	C7	05	AC	49	52	00	00	00	00	00	B8
00401170	FB	FF	FF	FF	C2	08	00	CC	CC	CC	CC	CC	CC	CC	CC	CC
00401180	6A	04	68	00	10	00	00	68	68	BE	1C	00	6A	00	FF	15
00401190	9C	63	52	00	50	FF	15	C8	63	52	00	8B	4C	24	04	6A
004011A0	00	6A	40	68	68	BE	1C	00	50	89	01	FF	15	C4	63	52

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.

```
text:00401000
text:00401000 > > > > >
text:00401000 > > > > >
text:00401000 > > > > >
text:00401000 > > > > >
text:00401000 > > > > >
text:00401000 56 > > > > >
text:00401001 8D 44 24 08 > > > > >
text:00401005 50 > > > > >
text:00401006 8B F1 > > > > >
text:00401008 E8 1C 1B 00 00 > > > > >
text:0040100D C7 06 08 BB 42 00 > > > > >
text:00401013 8B C6 > > > > >
text:00401015 5E > > > > >
text:00401016 C2 04 00 > > > > >
text:00401016 > > > > >
text:00401019 CC CC CC CC CC CC > > > > >
text:00401020 C7 01 08 BB 42 00 > > > > >
text:00401026 E9 26 1C 00 00 > > > > >
text:00401026 > > > > >

; Segment type: Pure code
; Segment permissions: Read/Execute
_text segment para public 'CODE' use32
    assume cs:_text
    ;org 401000h
    assume es:nothing, ss:nothing, ds:_data,
    push esi
    lea eax, [esp+8]
    push eax
    mov esi, ecx
    call ??0exception@std@@QAE@ABQBD@Z ; std.
    mov dword ptr [esi], offset off_42BB08
    mov eax, esi
    pop esi
    retn 4

; -----
    align 10h
    mov dword ptr [ecx], offset off_42BB08
    jmp sub_402C51
; -----
```

Target Value: Malware “class”

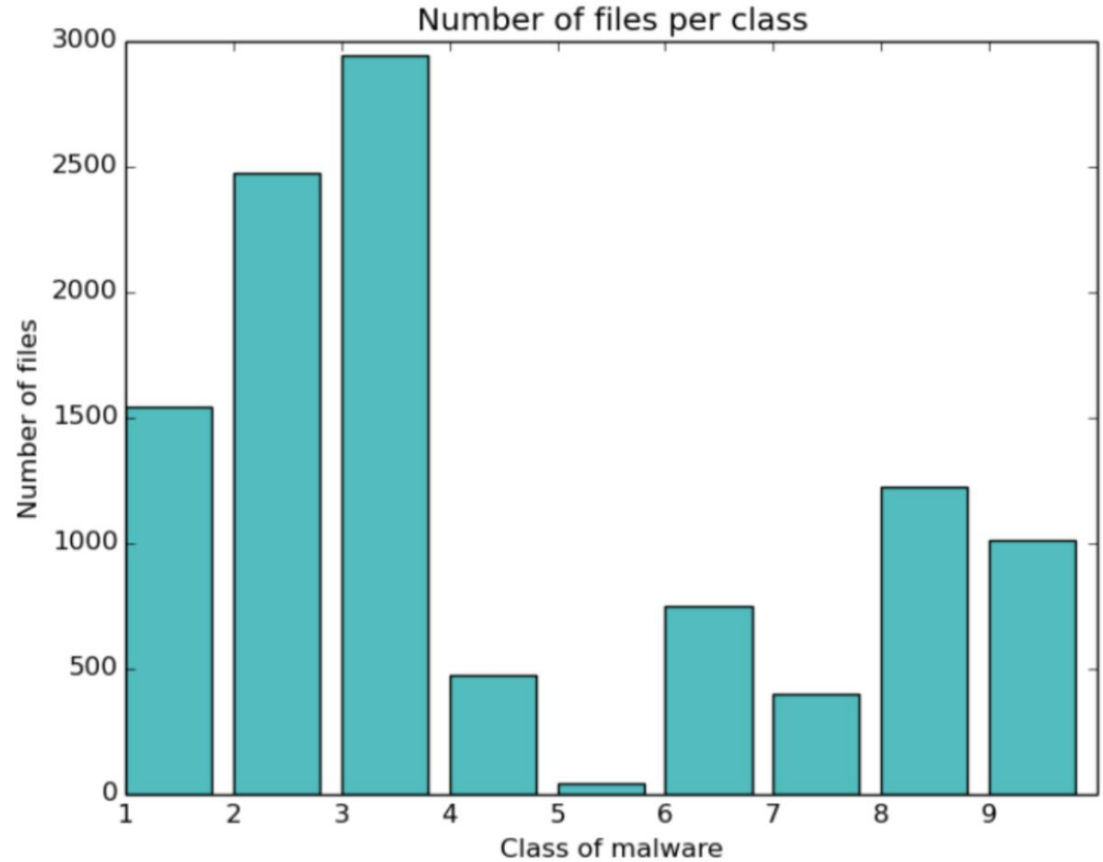
We were given the challenge to predict the class or “family” the test malware files belonged to with the highest accuracy.

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

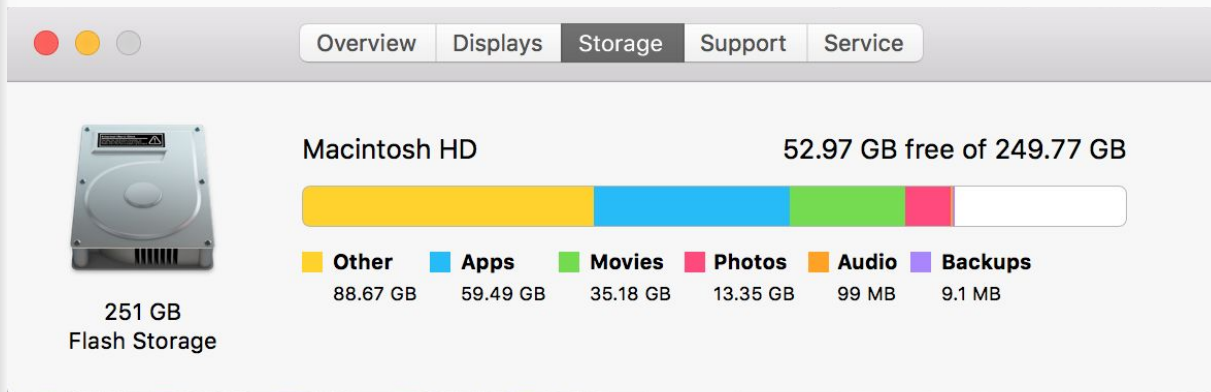
We tried to create the best model which gives us the highest accuracy on the test dataset. We tried the following classification models:

- Extra Trees Classifier
- LightGBM

Model training & techniques

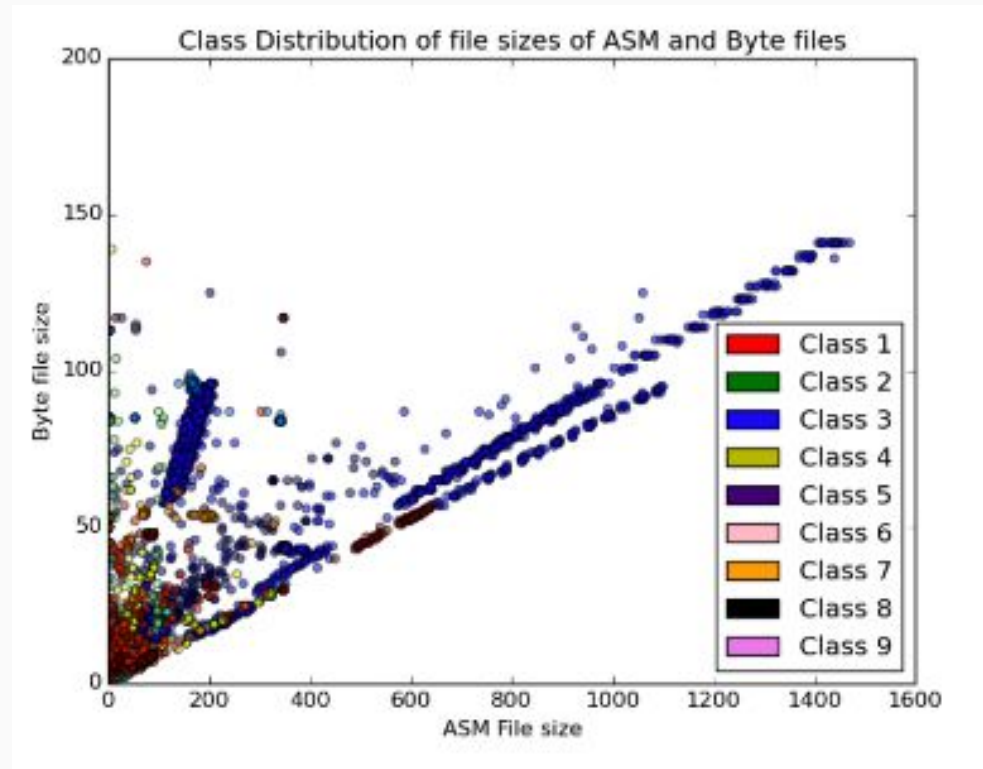
Dealing with Large Dataset

- 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

```
_generator(self, generator, val_samples, max_q_sl
1776                                     for out in outs:
1777                                     shape = (val_samp
-> 1778                                     all_outs.append(n
1779
1780                                     for i, out in enumerate(o

MemoryError:
```


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!

