Malware Classification into Families based on File - Content and Characteristics

KARAN BANSAL – 12342
PALAK AGARWAL – 13453

Motivation

- One of the major challenges faced by anti-malware today is the vast amount of data and files which needs to be evaluated for potential malicious content.
- Tens of millions of data points are generated daily to be analyzed as potential malware.
- Malware authors use automated techniques like Polymorphism in order to evade ‘pattern matching’ detection.
- Malware must be defined semantically as the same Virus, Worm, Trojan, Key Logger etc. is likely to exist in different physical forms.

Polymorphic Malware

- Polymorphism loosely means – ‘change the appearance of’.
- Spyware which constantly changes (‘morphs’) itself, making it difficult to detect with anti-malware programs.
- Generates a unique instance of a malware family for each victim, to create new malware.
- Evolution of malicious code can occur in a variety of ways such as filename changes, compression and encryption with variable keys.

Problem Statement and Challenge

- Training the classifier using the training data and then classifying the malware files (binary executables) in the test data into 9 categories of malwares.
- Identifying the classifying features in the byte code as well as asm file for each malware into their respective classes.
- Dataset is too large as compared to available computation power and resources.
- Appearance of malware (code) is different in every file making it difficult to identify common features of each class.

Data Set

- Participating in Microsoft Malware Challenge and the training as well as test dataset is provided by Kaggle.
- For every binary – byte code and disassembled asm file.
- Training set – 200 GB (10.8k asm files and 10.8k bytes files)
- Test set – 200 GB (10.8k asm files and 10.8k bytes files)
- Asm file – (0.4 millions – 19 millions lines)
- Bytes file – (150k - 180k lines)

Methodology

- Random Forest Classifier
- SVM
- Naïve-Bayes Classifier
- K-Nearest Neighbors
- N-gram based File Signatures
- K-Fold Cross Validation

Proposed Features

- Frequency of 256 possible hex values in the bytes file corresponding to each malware.
- Frequency of 256 possible hex values at specific position in the asm file corresponding to each malware.
- Frequency of various instructions like mov, jmp etc. in the asm file corresponding to each malware.
- N-gram based File Signatures

```
004010E0 00 FF 35 08 30 40 00 3E C3 58 FF D0 53 FF B3 88
004010F0 00 00 00 FF 73 60 FF B3 D0 00 00 00 FF B3 FC 02
00401100 00 00 E8 BD 06 00 00 83 C4 10 5B BF 30 30 40 00
00401110 8B 3F 57 FF B3 7C FF FF FF 5E FF D6 83 C4 28 5D
00401120 C3 00 00 00 8B FF 55 8B EC 83 EC 34 8D 91 FC 00
00401130 00 00 81 FA 65 22 00 00 74 ED 52 BB 00 00 00 00
00401140 53 E8 08 00 00 00 FF 35 4C 30 40 00 3E C3 58 FF
00401150 D0 5A 52 50 50 53 51 E8 28 07 00 00 83 C4 10 5A
00401160 52 BF 00 00 00 00 57 E8 08 00 00 00 FF 35 34 30
00401170 40 00 3E C3 58 FF D0 5A 83 C4 34 5D C3 00 00 00
00401180 00 00 00 00 00 00 00 00 8B FF 55 8B EC 83 EC 24
00401190 8D B2 E0 01 00 00 1B BE 6C FD FF FF 56 52 50 FF
004011A0 B6 B4 01 00 00 E8 5E 0C 00 00 83 C4 0C 5E BA 00
004011B0 00 00 00 52 E8 08 00 00 00 FF 35 7C 30 40 00 3E
004011C0 C3 58 FF D0 B9 00 00 00 00 51 E8 08 00 00 00 FF
004011D0 35 28 30 40 00 3E C3 58 FF D0 83 C4 24 5D C3 00
004011E0 8B FF 55 8B EC 83 EC 48 8D 1D 7C D1 57 00 09 9B
004011F0 60 FF FF FF BF 00 00 00 00 57 E8 08 00 00 00 FF
00401200 35 78 30 40 00 3E C3 58 FF D0 53 FF B3 B4 02 00
00401210 00 52 FF B3 3C 02 00 00 FF B3 54 03 00 00 E8 09
00401220 06 00 00 83 C4 10 5B 6A 00 E8 08 00 00 00 FF 35
00401230 20 30 40 00 3E C3 58 FF D0 83 C4 48 5D C3 00 00
00401240 8B FF 55 8B EC 83 EC 60 8D 8A 7C FE FF FF 11 4D
00401250 E4 51 6A 00 E8 08 00 00 00 FF 35 78 30 40 00 3E
00401260 C3 58 FF D0 59 51 50 50 56 E8 CA 09 00 00 83 C4
00401270 0C 59 51 BA 00 00 00 00 52 E8 08 00 00 00 FF 35
00401280 38 30 40 00 3E C3 58 FF D0 59 83 C4 60 5D C3 00
00401290 00 00 00 00 8B FF 55 8B EC 83 EC 54 8D BB 8C FE
```

```
                loc_41DBED:                     ; CODE XREF: sub_41DB8F+56CANj
F FF 1E                          mov     byte ptr [ebp+var_25+2], 1Eh
                        cmp     edx, edi
                        js      short loc_41DBFE
                            add     [ebp+var_14], ebx
                            xor     [ebp-26h], ebx

                loc_41DBFE:                     ; CODE XREF: sub_41DB8F+67CANj
                            push    [ebp+var_2B]
                            and     eax, [ebp+var_25]
                        jnz     short loc_41DC0C
                            and     [ebp+var_10+2], eax
                            sub     [ebp+var_2B+3], esi

                loc_41DC0C:                     ; CODE XREF: sub_41DB8F+75CANj
                        add     edx, esp
F FF B6                          mov     [ebp+var_1C], 0B6h
                            and     [ebp+var_21], edx
                            lea     edx, [esi+eax*4+21h]
C 00 00                          lea     edx, [ebx+eax*4+2C57h]
                            add     esp, 10h
                        pop     eax
                        leave
                        retn
                sub_41DB8F      endp
```

```
.data:00638087 F8                                   db 0F8h ; ø
.data:00638088 A3                                   db 0A3h ; £
.data:00638089 B4                                   db 0B4h ; ´
.data:0063808A 24                                   db  24h ; $
.data:0063808B 99                                   db  99h ; ™
.data:0063808C B2                                   db 0B2h ; ²
.data:0063808D 79                                   db  79h ; y
.data:0063808E 2F                                   db  2Fh ; /
.data:0063808F F0                                   db 0F0h ; ð
.data:00638090 BD                           unk_638090      db 0BDh ; ½         ; DATA XREF: sub_400E58+8CANo
.data:00638090                                                  ; sub_635A58+8CANo
.data:00638091 34                                   db  34h ; 4
.data:00638092 4B                                   db  4Bh ; K
.data:00638093 9E                                   db  9Eh ; ž
.data:00638094 7D                                   db  7Dh ; }
.data:00638095 A7                                   db 0A7h ; §
.data:00638096 30                                   db  30h ; 0
.data:00638097 40                                   db  40h ; @
.data:00638098 12                                   db  12h
.data:00638099 4F                                   db  4Fh ; O
.data:0063809A 88                                   db  88h ; ˆ
.data:0063809B B2                                   db 0B2h ; ²
.data:0063809C 03                                   db    3
.data:0063809D 7E                                   db  7Eh ; ~
.data:0063809E 6F                                   db  6Fh ; o
.data:0063809F B2                                   db 0B2h ; ²
.data:006380A0 AE                                   db 0AEh ; ®
.data:006380A1 13                                   db  13h
.data:006380A2 1B                                   db  1Bh
.data:006380A3 E9                                   db 0E9h ; é
.data:006380A4 79                           unk_6380A4      db  79h ; y         ; DATA XREF: sub_400EE0+8CANo
.data:006380A4                                                  ; sub_635AE0+8CANo
.data:006380A5 FC                                   db 0FCh ; ü
.data:006380A6 1D                                   db  1Dh
.data:006380A7 38                                   db  38h ; 8
.data:006380A8 3D                                   db  3Dh ; =
.data:006380A9 85                                   db  85h ; …
.data:006380AA 61                                   db  61h ; a
```

Submission and Score Calculation

- For each malware file we'll submit a set of predicted probabilities : (one for every class)
- Each file has been labelled with one true class.
- Evaluation is done using Multi-Class Logarithmic Loss.
- $$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij}),$$
- Minimize the log loss to achieve higher accuracy.

Current Progress

- Applied Random Forest Classifier on bytes files with frequency of 256 hex values as features achieving a score of 0.1929345.
- Applied Random Forest Classifier on asm files and code is running on the machines.
- Explored the asm and bytes files and figured out some distinguishing patterns in malwares corresponding to nine families.

* Code of random forest classifier taken from Vishnu Chevli (github.com/vrajs5/Microsoft-Malware-Classification-Challenge).

REFERENCES :

- Bilar, Daniel. "Statistical structures: Fingerprinting malware for classification and analysis." Proceedings of Black Hat Federal 2006 (2006).
- Griffin, Kent, et al. "Automatic generation of string signatures for malware detection." Recent Advances in Intrusion Detection. Springer Berlin Heidelberg, 2009.
- Santos, Igor, et al. "N-grams-based File Signatures for Malware Detection."ICEIS (2) 9 (2009): 317-320.
- Raman, Karthik. "Selecting features to classify malware." InfoSec Southwest(2012).

Thank You

Any Questions?