ISSN 1561-5359. Штучний інтелект, 2018, № 3

УДК 004.93

A.A. Voranau[1], Y.V. Harakhavik[2]
[1]United Institute of Informatics Problems NAS of Belarus, Belarus
6, Syrganov st., Minsk, 220013
[2]Belarusian State University of Informatics and Radioelectronics, Belarus
6, P. Brouki st., Minsk, 220013

MACHINE LEARNING APPROACH FOR MALWARE DETECTION USING EXECUTABLE FILES FEATURES EXTRACTION

А.А. Воронов[1], Я.В. Гороховик[2]
[1]Государственное научное учреждение «Объединенный институт проблем информатики Национальной академии наук Беларуси», Беларусь
ул. Сурганова, 6, г. Минск, 220013
[2]УО «Беларусский государственный Университет информатики и радиоэлектроники», Беларусь
ул. П. Бровки, 6, г. Минск, 220013

МЕТОД МАШИННОГО ОБУЧЕНИЯ ДЛЯ ДЕТЕКТИРОВАНИЯ ВРЕДОНОСНОГО ПО, ИСПОЛЬЗУЮЩИЙ ИЗВЛЕЧЕНИЕ ПРИЗНАКОВ ИЗ ИСПОЛНЯЕМЫХ ФАЙЛОВ

A malicious software is generally an executable program which usually settles itself in the system, replicates by copying itself, and has a malicious effect. Modern antivirus systems detect malware by knowing its pattern and detect a new virus quite difficult. There are a lot of heuristic techniques are used for detecting an unknown malware which are usually consume a lot of system memory and CPU resources. This load can be overcome by training a machine learning model which collects features from Portable Executable (PE) file which are used for identifying an unknown virus patterns. A technique to collect these features from PE file is proposed in this paper.

Keywords: Malware, Machine learning, Heuristics, PE-files

Вредоносное ПО, как правило, представляет собой исполняемую программу, которая обычно располагается в системе, реплицируется путем копирования и оказывает вредоносное воздействие. Современные антивирусные системы обнаруживают вредоносное ПО, зная его паттерн, а обнаруживать новый вирус довольно сложно. Существует множество эвристических методов, используемых для обнаружения неизвестных вредоносных программ, которые обычно потребляют много системной памяти и ресурсов процессора. Эту нагрузку можно преодолеть путем обучения модели машинного обучения, которая собирает данные из Portable Executable (PE) файла, которые используются для идентификации неизвестных вирусных паттернов. В данной статье предлагается метод сбора этих характеристик из PE-файла.

Ключевые слова: Вредоносные программы, Машинное обучение, Эвристика, PE-файлы

Introduction

Nowadays one of the main problems of the informational security is the identifying new malicious software and threats. Known viruses do not pose a particular danger since they are easily detected by hash analysis. But detection of new threats requires advanced heuristic methods. There are several ways to identify such threats:

1. Reveal similarities between malware families by focusing on the biggest malware groups. It usually based on such machine learning (ML) algorithms as Bayesian networks or genetic algorithms. The input file produces several features which are compared with known virus patterns.
2. Implement a set of algorithms that emulate the decision-making strategy of a human analyst. A human malware analyst can determine that a Windows PE program appears malicious, without actually observing its behavior, by briefly analyzing the file structure and taking a quick look at the disassembly of the file. The analyst would be asking the following questions: Is the file structure uncommon? Is it using tricks to fool a human? Is the code obfuscated? Is it using any anti-debugging tricks? If the

answer to such questions is “yes”, then a human analyst would suspect that the file is malicious.

3. Analyze a suspected file in a “sandbox” which require implementing User-mode and kernel-mode hooks. This approach allows to execute a suspected file in a virtual environment to look for suspicious activities. It means that we can observe the real behavior of the executable file [1].

Limitations of the existing approaches

As described, there are several approaches for detecting an unknown malware. However, each of the approaches has its limitations:

1. The first method can cause a large number of false positives and consume a lot of resources, which is acceptable in malware research lab environment and is not suitable for desktop solutions.
2. The second method is more reliable than any other approach because it involves actually looking at the true runtime behavior. However, it’s too complex and consumes a lot of system memory and CPU resources.
3. The third method largely depends on the quality of the corresponding CPU emulator engine and the quality of emulated operation system APIs. Even if it is effective it is time consuming and costly.

The proposed approach

This paper describes the method of PE-files features extraction to determine if the file is malicious and the performance evaluation of this method. The method was tested on unpacked Windows x86 executable files.

The main approach allows determining what is the suspected file supposed to do by collecting the following features:

1. Common features – features of the file itself.
2. PE structure features – features of PE header and Import table.
3. Code features – features taken from disassembler listing.
4. Behavioral features – most common patterns for the malware.

A virus usually needs to settle itself in the system. It means it should use functions for accessing and editing the Windows registry. Also it can copy itself to system directories which means it has to use functions for escalating privileges and coping a file. After a malware is settled in the registry and spread itself over the system it can access the remote host for requesting a command from it. It can be done by calling networking functions.

All these functions are combined such a way that we can likely predict the behavior of the analyzed file. Since we know from PE header the exact location of IAT and because of its relatively small size we can collect the functions combination set of a suspected file without consuming a lot of OS resources. A neural network which was trained with a set of functions combination of malicious and legitimate files can determine a supposed behavior of the suspected file without actually executing it.

Moreover, a PE file contains a lot of data which helps to determine if the file was edited. We can check several values such as sizes of code and data sections, the offset of Original Entry Point (OEP), the Relative Virtual Address (RVA) of PE Image, the RVA and size of Import Address Table (IAT), Export Address Table (EAT) and Resource Table.

These features can prove us that the file was patched in case if its OEP was rewritten or it has a certain section size for one particular virus.

The most of code and behavioral features are non-numerical. The machine learning model of our approach needs data in numerical form but we can’t encode these features as numbers because they won’t be true categorical features. Instead these features will be converted into a separate binary feature that has value 1 for instances for which the category appeared and value 0 when it didn’t. Hence, each categorical feature is converted to a set of binary features, one per category. [2]

Common features include three values:

1. File type (DLL, console, GUI, native) – categorical feature.

2. File entropy – may point that the file contains en encrypted content.
3. File size.

PE structure features include Import table (IAT) features and the values of PE header fields.

For collecting the IAT features the set of 96 Win32 API functions was created. There are functions which are commonly used by malware as well as antiviruses such as AdjustTokenPrivileges, CreateRemoteThread, GetProcAddress, VirtualProtectEx, WriteProcessMemory and a lot of others.

Based on statistics from 46000 malicious programs only 56 functions were kept. These most popular functions are used as categorical features.

Code features include the periodicity of CPU registers and instructions using. Control flow graph features are also related to code features: vertex count, edge count, delta max and density.

The periodicity of registers using can help detecting different malicious technics such as a current virtual address revealing which was a well-known technic of file viruses. This technic is illustrated in the Figure 1.

```
call    $+5
pop     eax
```

Fig. 1. Current RVA revealing

When a malware uses register for relative addressing like in the example above or for storing the address of a dynamically load library the periodicity of using this register usually grows or falls respectively.

Behavioral features represent the popular behavior patterns which is typically used by the malware. The features are collected by parsing the disassembler listing. The most popular 20 malicious technics were selected: 1) Current RVA revealing; 2) VirtualAlloc with RVA rights; 3) WriteProcessMemory to the current process memory; 4) WriteProcessMemory to the remote process memory; 5) DLL injection; 6) Keylogger routine; 7) Registry modification; 8) WinInet API using; 9) PEB address obtaining; 10) Process replacement; 11) User Mode APC injections; 12) Process Hollowing; 13) Thread Execution Hijacking; 14) SetWindowsHookEx using; 15) Extra Window memory Injection (EWMI); 16) Inline Hooking; 17) Kernel Mode APC hooking; 18) SSDT hooking; 19) IDT hooking; 20) SYSENTER/SYSCALL hook.

Features collection

The basic idea of Fisher Score is to find a subset of features of the data such that in the data space spanned by selected features, distance between data points in different classes are as large as possible and distance between data points in the same class are as small as possible. Fisher score computes the difference, in terms of mean and standard deviation, between positive and negative examples relative to a particular feature. It assigns ranks to each feature. Rank of a feature is defined as the ratio between absolute difference between the means of positive and negative examples and the sum of the standard deviations of the positive and negative examples, when considering that feature [3-11]. A large value of a rank implies greater difference in positive and negative examples, considering that feature, hence is more important for separating positive and negative values. Thus, this feature is relevant. A small value of rank would imply a lesser difference in positive and negative examples, hence is less important for separating positive and negative values [12-15]. Thus, this feature is irrelevant.

$$R_i = \frac{\left|\mu_{i,p} - \mu_{i,n}\right|}{\sigma_{i,p} + \sigma_{i,n}} \qquad (1)$$

where R_i – the rank of i feature, $\mu_{i,p}$ and $\mu_{i,n}$ – the mean of legitimate and malicious examples features correspondently, $\sigma_{i,p}$ and $\sigma_{i,n}$ – the standard deviations.

The Fisher Score approach was applied to PE header features and Code features. Figures 2 and 3 show Fisher ranks for these groups of features.

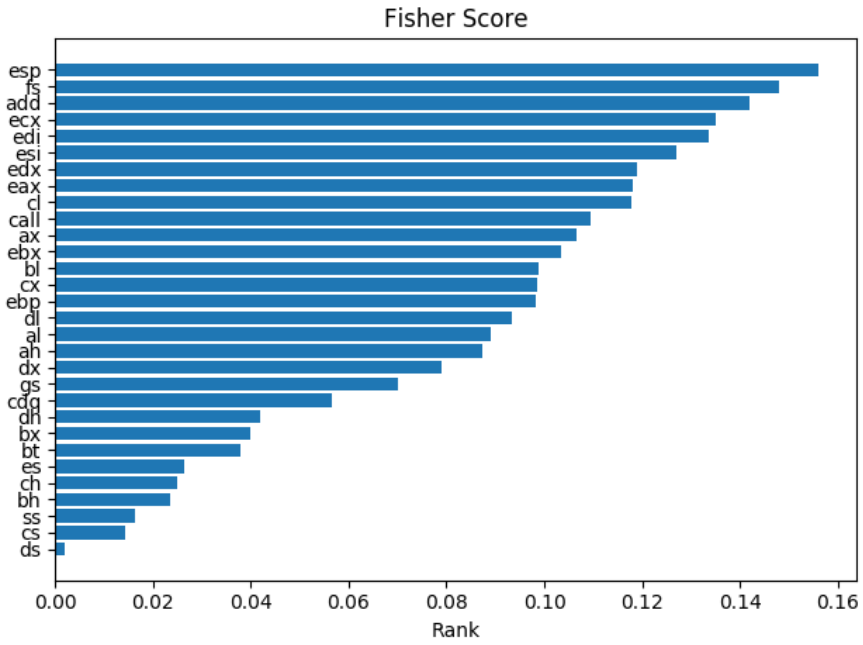

Fig. 2. 30 code features with highest ranks

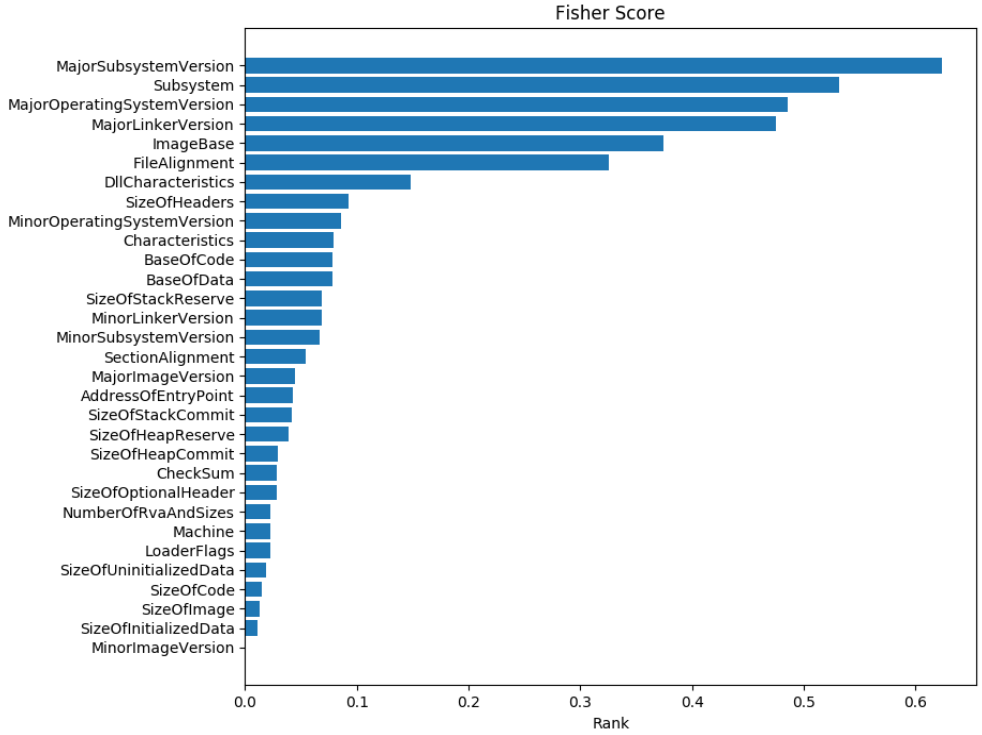

Fig. 3. Ranks of PE header features

Only 29 code features were left from 119 in total. PE header features number were reduced to three. The selected PE header features are ImageBase, FileAlignment, DllCharacteristics.

Model testing

The training set consists of 26628 malicious samples and 9115 legitimate samples. The testing set consists of 9115 malicious and 814 legitimate samples.

The model was trained and tested using five machine learning algorithms: 1) Decision Tree (DT); 2) «Random Forest» (RF); 3) Gradient Boosting (GB); 4) Adaptive Boosting (AB); 5) Naive Bayes classifier (GNB).

The results of the model testing are shown in the Table 1.

Table 1. Classification results

	T_m	R_m	T_l	R_l	FP	FN
GNB	1410	1839	814	385	30.4%	11.1%
DT		1085		1139	18.6%	33.2%
RF		859		1365	13.4%	38.1%
AB		1593		631	19.9%	11.7%
GB		1830		394	27.1%	8.2%

T_m and T_l are numbers of malicious and legitimate sample in the testing set. R_m and R_l are the number of samples that were detected as malicious and legitimate respectively. FP and FN are False-positive and False-negative rates.

The best result was shown by Gradient Boosting algorithm. Only 8.2% of malicious samples were classified as legitimate.

Conclusion

In this paper, the feature selection approach for PE files was described. The approach includes the selection of static features (such as PE header fields values and file entropy) and behavioral features (the popular malicious patterns).

The described approach was applied to a set of malicious and legitimate samples to demonstrate its efficiency.

The model optimization and classifier improvements can be executed in further.

References

1. Koret, J., Bachaalany, E. (2015) *The Antivirus Hacker's Handbook*. Indianapolis, Indiana: John Wiley & Sons, Inc.
2. Brink, H., Richards, J.W., Fetherolf, M. (2017) Real-World Machine Learning. Shelter Island, NY: Manning Publications Co.
3. Stopel, D., Boger, Z., Moskovitch, R., Shahar, Y., Elovici, Y. (2006) *Improving Worm Detection with Artificial Neural Networks through Feature Selection and Temporal Analysis Techniques*. Be'er Sheva, Israel: Deutsche Telekom Laboratories at Ben-Gurion University.
4. Nikolenko, S., Kadurin, A., Arhangelskaya, E. (2018) Glubokoe obuchenie – SPb.: Piter. – 480 s.
5. Tensorflow API documentation [Electronic resource]. Available from: https://www.tensorflow.org/api_docs/python/tf/nn/softmax_cross_entropy_with_logits_v2. Date of access: 28.09.2018.
6. Sikorski, M. (2012) Practical Malware Analysis. San Francisco: No Starch Press, Inc. 800 p.
7. Kasperski, K. (2005) Zapiski issledovatelya kompyuternyih virusov. – SPb.: Piter. 316 s.
8. Ligh, M.H. (2011) Malware Analyst's Cookbook and DVD: Tools and Techniques for Fighting Malicious Code. Indianapolis: Wiley Publishing, Inc. 744 p.
9. Marak, V. (2015) Windows Malware Analysis Essentials: Master the fundamentals of malware analysis for the Windows platform and enhance your anti-malware skill set. Birmingham: Packt Publishing Ltd. 330 p.
10. Shamir, A. (1985) Identity-based cryptosystems and signature schemes. Advances in cryptology, Springer, pp. 47–53.
11. Sahai, A., Waters, B. (2005) Fuzzy identity-based encryption. *Advances in Cryptology–EUROCRYPT 2005*, pp. 557–557.
12. Wolf, S. (1998) Unconditional Security in Cryptography. Lectures on Data Security, Modern Cryptology in Theory and Practice, volume 1561 of Lecture Notes in Computer Science, pp. 217–250. Springer-Verlag, July 1998.
13. Cappaert, J., Wyseur, B., Preneel, B. (2004) Software security techniques. COSIC internal report, Katholieke Universiteit Leuven.
14. Dent, A.W. (2006) Fundamental problems in provable security and cryptography. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 364(1849):3215–3230.
15. Kasperski, K. (2008) Iskusstvo dizassemblirovaniya. BHV.: Piter, 896 s.

РЕЗЮМЕ

А.А. Воронов, Я.В. Гороховик

Метод машинного обучения для детектирования вредоносного ПО, использующий извлечение признаков из исполняемых файлов

В данной работе описывается метод извлечения различных признаков из исполняемых файлов с целью обучения модели машинного обучения для детектирования вредоносного программного обеспечения.

В настоящее время одной из основных проблем информационной безопасности является выявление новых видов вредоносного программного обеспечения. И если уже известные вредоносные программы не представляют особой опасности, так как легко определяются с помощью сигнатурного анализа, то для обнаружения новых, ранее не выявлявшихся, угроз используются более продвинутые эвристические методы.

Огромное количество эвристических методов, используемых для детектирования вредоносного программного обеспечения различными антивирусными продуктами, обычно потребляют значи-

тельное количество ресурсов электронной вычислительной машины и процессора. Данная нагрузка может быть снижена за счёт обучения нейронной сети, которая, на основании собранных признаков исполняемых файлов, определяет шаблоны поведения вирусов.

Предлагаемый метод заключается в извлечении статических и динамических признаков без исполнения файла и в определении наиболее весомых из них с помощью критерия Фишера и соответствует следующим требованиям:

1. Для анализа поведения программы требуется только её исполняемый файл. Не предполагается запуск программы для его динамического анализа с помощью виртуальных машин, API-логгеров и дополнительного программного обеспечения.
2. Анализ файла осуществляется на основании большого количества признаков, полученных непосредственно их исполняемого файла.
3. Признаки могут относиться как к структуре исполняемого файла, так и к его поведению.
4. При моделировании признаков предпочтение отдаётся численным, нежели категориальным.
5. Все категориальные признаки преобразуются в численные бинарные признаки, для того чтобы избежать случайности при индексировании категорий.
6. Для обучения и тестирования модели используются только «неупакованные» PE-файлы.

Описываемый метод подкреплён результатами тестирования пяти алгоритмов машинного обучения: Дерево решений, «Случайный лес», Градиентный и Адаптивный бустинг и Наивный байесовский классификатор.

Надійшла до редакції 25.10.2018

