Methods of information systems protection

UDC 004.492.34 **doi: 10.20998/2522-9052.2017.1.08**

S. Gavrilenko, D. Saenko

National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine

DEVELOPMENT OF THE METHOD AND PROGRAM MODEL OF THE STATIC ANALYZER OF HARMFUL FILES

The **subject** of research in this article is the methods of analyzing malicious software. The **goal** is to improve the secure functioning of computer systems (CS) and protect them from the effects of computer viruses. **Research target**: the research of modern means of software antivirus protection; analysis of the methods of creating a file signature; the development of a software model for static file detection, based on the analysis of the PE structure; the generation of tables of features that are inherent to families of viruses such as Worms, Backdor, Trojan; the obtainment binary signatures of malicious and secure software. The **methods** used are: analysis of the code in a Hex file, file hashing algorithms. The following **results** are obtained. The PE-structure of the file has been analyzed; sections have been selected for further analysis. A software model of static file detection has been developed and the analysis of secure and malicious files has been performed. Features in the form of strings and API functions have been selected; a bitmask has been formed for further file analysis. 3500 files of malicious and safe software has been scanned, their analysis has been performed. Signatures of each malicious file have been encoded and stored in the signature database. Using the developed software model, a study has been made of the possibility of detecting modifications to malicious software. **Conclusions**. A method and software model of static detection of malicious files has been developed, which allow automatic obtainment of a set of file features and draw a conclusion about the severity of the file.

Keywords: malicious software, signature, Python, portable execute, malicious application, API functions, harmful files.

Introduction

The times when the information security was reduced to the policies and protection of all devices in the corporate network are a thing of the past. Today this is clearly not enough. Cyber threats are developing rapidly, and the understanding of which direction this development is taking place plays a key role in ensuring the effective protection of enterprises [1]. If the viruses were not detected at an early stage, the recovery cost after an attack increases more than twofold. For example, the total recovery cost after a cyber attack lasting a week or more is over 1 million dollars. At the same time, the immediate reaction to the malfunction costs the company an average of 400 thousand dollars.

To date, there is a great number of anti-virus programs, but they are not capable of completely protecting the information stored on the computer, so a timely detection of malicious software is a crucial task.

Analysis of the problem and formulation of the research target

The analysis of the literature [2-8] has proven that many specialized anti-virus programs are used as protection from cyber attacks, whose work is most often based on the technologies of signature and heuristic analysis. One of the components of suspicious software analysis is static detection – according to the file analysis conducted in binary format and dynamic detection – according to their behavior in the system [9, 10].

Threat data is collected from a variety of sources, including cloud infrastructure, web crawlers, botnet monitoring services, spam traps. New cyber threats are determined by checking URLs, domains, IP addresses, file checksums, timestamps, file names, DNS data, and other features that are inherent to the programs. The received information is thoroughly checked, systematized, cleaned and analyzed both by technical means and by company analysts that are developing the antivirus software.

At the same time, to date, there are no automated systems of decision-making on the account of file severity and the building of a signature for newly detected malicious software.

The solution for the research target

For the analysis of files, two types of searches are used for detecting anomalies: static and dynamic [12-16]. Static code analysis is based on the analysis of the frequency of using the processor's commands and on the basis of this information a conclusion is made concerning the file's virus infection.

The main demerit of this method is that there is a number of complex polymorphic viruses that use almost all the processor commands and from copy to copy the set of used commands varies greatly, therefore, according to the constructed frequency table it is not possible to detect the virus.

The method of dynamic code analysis consists in analyzing the executable code in a special "environment", called the emulation buffer or "sandbox". The result of this analysis is a summary of objects that were active during the execution of the file. A modern dynamic method can check not only the processor's commands, but also the activation of the operating system. The task of writing a full-fledged dynamic analyzer is quite laborious, not to mention the fact it requires constant monitoring of the actions of each command. This is necessary in order to not accidentally activate the destructive components of the virus algorithm.

In this paper, a program model for the static detection of files in binary format has been developed according to the analysis of the PE-structure of executable files in order to obtain features that are characteristic for malicious files [10,12,14]. The analyzed files are a family of viruses such as Worms, Backdor, Trojan. The developed software allows the analysis of the import and export sections of the file's structure and receives the names of functions and dynamic libraries, as well as information about function arguments: the names of the services used, the names of the processes to be deleted or created, various network peculiarities (IP addresses, ports, resource addresses, email addresses). The analysis of the PE-structure of the file made it possible to identify a number of parameters for further investigation. As a further study, it was decided to use the following parameters:

- Shannon's entropy of the data section; ($H=\sum_{i=0}^{N}(N_i/N)\cdot\log(N_i/N)$);
- a compiler or /packager;
- number of sections;
- availability of a certificate;
- the presence of a record during boot up;
- list of used API functions that cause suspicions in the executable file.

These parameters characterize:

- file compression;
- a packed file may raise more suspicion;
- a large number of sections can cause suspicion;
- the availibility of a certificate reduces the likelihood of file damage;
- the presence of a record during bootup causes increased attention;

As an example, 290 files of Worm type, 1050 files of Trojan type, 1153 files of Backdoor type, 1000 safe files have been analyzed in this paper. The application is written in Python with the use of pefile libraries and sqlite3 database.

The first stage of the study is the removal of information from the PE-structure of malicious software: and the search for API functions from the import and strings table (Hex-sequences of a given length, Fig. 1).

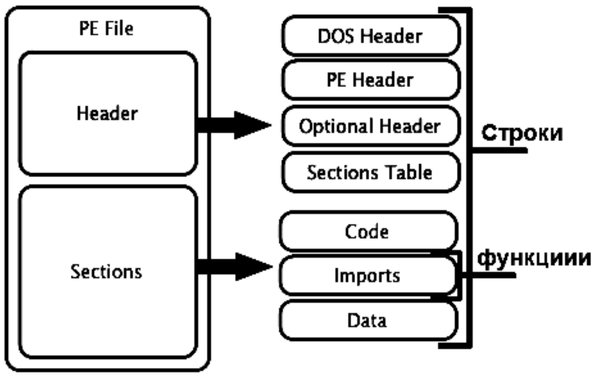
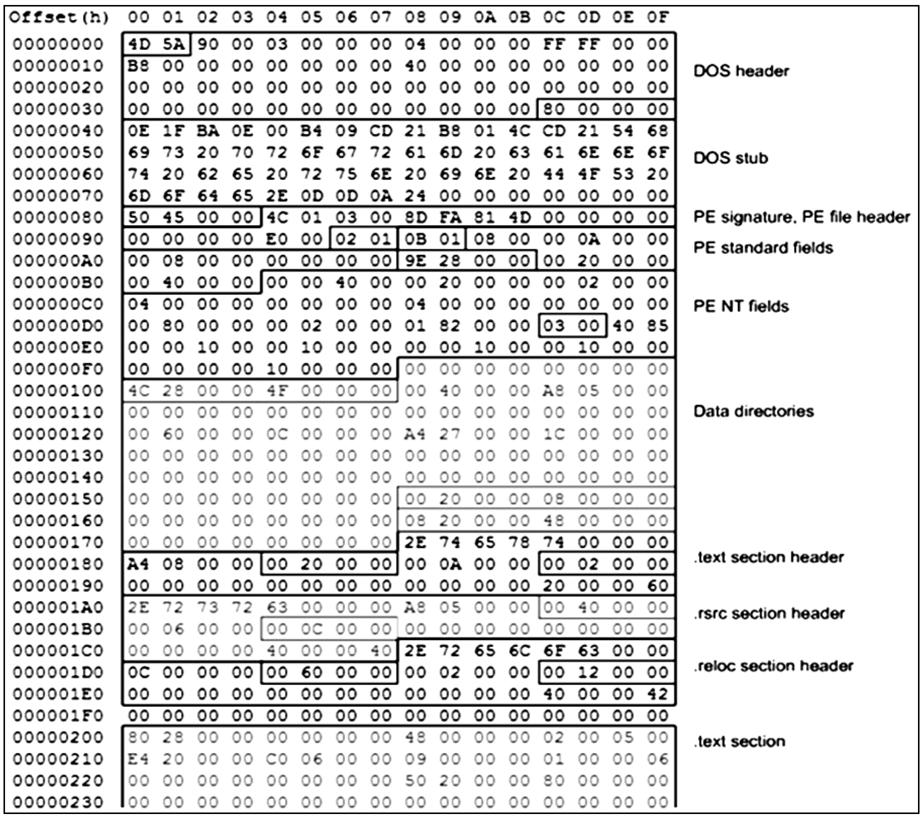


Fig. 1. PE structure of executed file

In Fig. 2 an example is shown of a PE file structure with highlighted areas of analysis.



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00
00000010  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   DOS header
00000020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000030  00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 00
00000040  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68
00000050  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F   DOS stub
00000060  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20
00000070  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00
00000080  50 45 00 00 4C 01 03 00 8D FA 81 4D 00 00 00 00   PE signature, PE file header
00000090  00 00 00 00 E0 00 02 01 0B 01 08 00 00 0A 00 00   PE standard fields
000000A0  00 08 00 00 00 00 00 00 9E 28 00 00 00 20 00 00
000000B0  00 40 00 00 00 00 40 00 00 20 00 00 00 02 00 00
000000C0  04 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00   PE NT fields
000000D0  00 80 00 00 00 02 00 00 01 82 00 00 03 00 40 85
000000E0  00 00 10 00 00 10 00 00 00 00 10 00 00 10 00 00
000000F0  00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00
00000100  4C 28 00 00 4F 00 00 00 00 40 00 00 A8 05 00 00
00000110  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   Data directories
00000120  00 60 00 00 0C 00 00 00 A4 27 00 00 1C 00 00 00
00000130  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000140  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000150  00 00 00 00 00 00 00 00 00 20 00 00 08 00 00 00
00000160  00 00 00 00 00 00 00 00 08 20 00 00 48 00 00 00
00000170  00 00 00 00 00 00 00 00 2E 74 65 78 74 00 00 00
00000180  A4 08 00 00 00 20 00 00 00 0A 00 00 00 02 00 00   .text section header
00000190  00 00 00 00 00 00 00 00 00 00 00 00 20 00 00 60
000001A0  2E 72 73 72 63 00 00 00 A8 05 00 00 00 40 00 00   .rsrc section header
000001B0  00 06 00 00 00 0C 00 00 00 00 00 00 00 00 00 00
000001C0  00 00 00 00 40 00 00 40 2E 72 65 6C 6F 63 00 00
000001D0  0C 00 00 00 00 60 00 00 00 02 00 00 00 12 00 00   .reloc section header
000001E0  00 00 00 00 00 00 00 00 00 00 00 00 40 00 00 42
000001F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000200  80 28 00 00 00 00 00 00 48 00 00 00 02 00 05 00   .text section
00000210  E4 20 00 00 C0 06 00 00 09 00 00 00 01 00 00 06
00000220  00 00 00 00 00 00 00 00 50 20 00 00 80 00 00 00
00000230  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

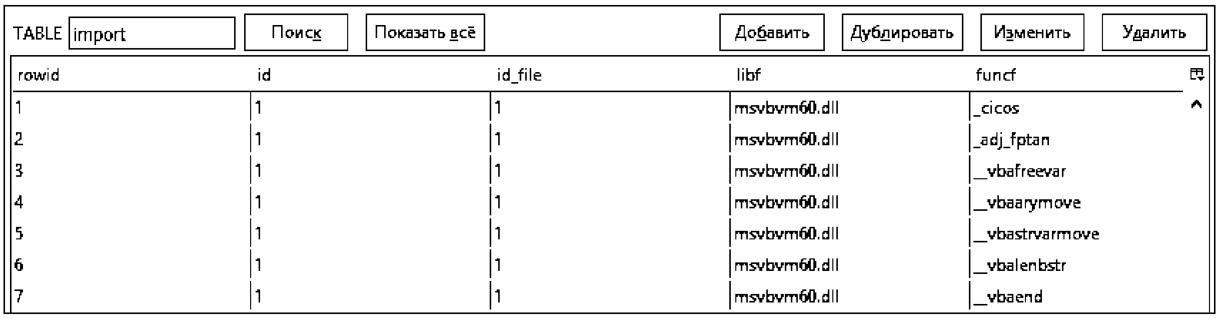
Fig. 2. Example of a PE file structure

The result of the analysis of the PE malicious files structure being investigated is presented in two tables:

- a table with API-functions and libraries, in which they are included. A total of 24,945 entries were received (Fig. 3).

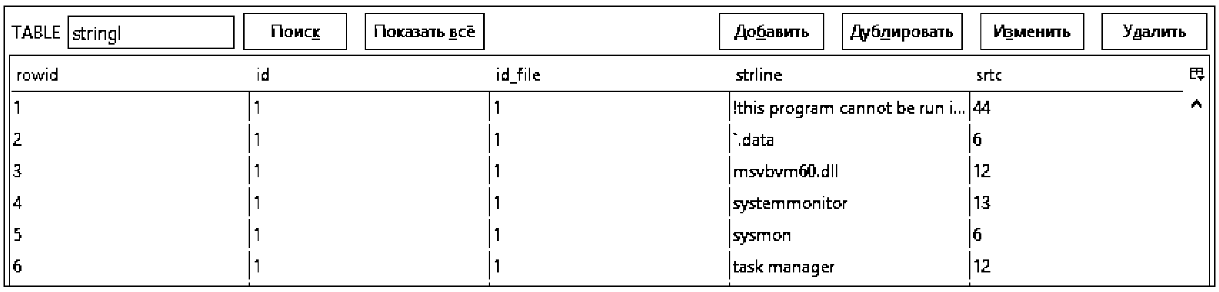
- table with strings (Fig. 4). A total of 175651 strings were found, the length of which varied from 6 to 70 characters.

In a similar manner, the testing of secure software has been performed. A fragment of the test results is shown in Fig. 5.



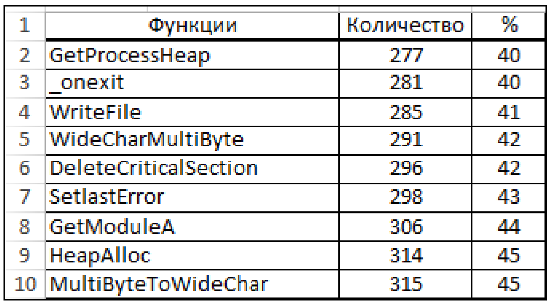
rowid	id	id_file	libf	funcf
1	1	1	msvbvm60.dll	_cicos
2	1	1	msvbvm60.dll	_adj_fptan
3	1	1	msvbvm60.dll	__vbafreevar
4	1	1	msvbvm60.dll	__vbaarymove
5	1	1	msvbvm60.dll	__vbastrvarmove
6	1	1	msvbvm60.dll	__vbalenbstr
7	1	1	msvbvm60.dll	__vbaend

Fig. 3. Table of found libraries and API-functions



rowid	id	id_file	strline	srtc
1	1	1	!this program cannot be run i...	44
2	1	1	`.data	6
3	1	1	msvbvm60.dll	12
4	1	1	systemmonitor	13
5	1	1	sysmon	6
6	1	1	task manager	12

Fig. 4. Table of found strings



1	Функции	Количество	%
2	GetProcessHeap	277	40
3	_onexit	281	40
4	WriteFile	285	41
5	WideCharMultiByte	291	42
6	DeleteCriticalSection	296	42
7	SetlastError	298	43
8	GetModuleA	306	44
9	HeapAlloc	314	45
10	MultiByteToWideChar	315	45

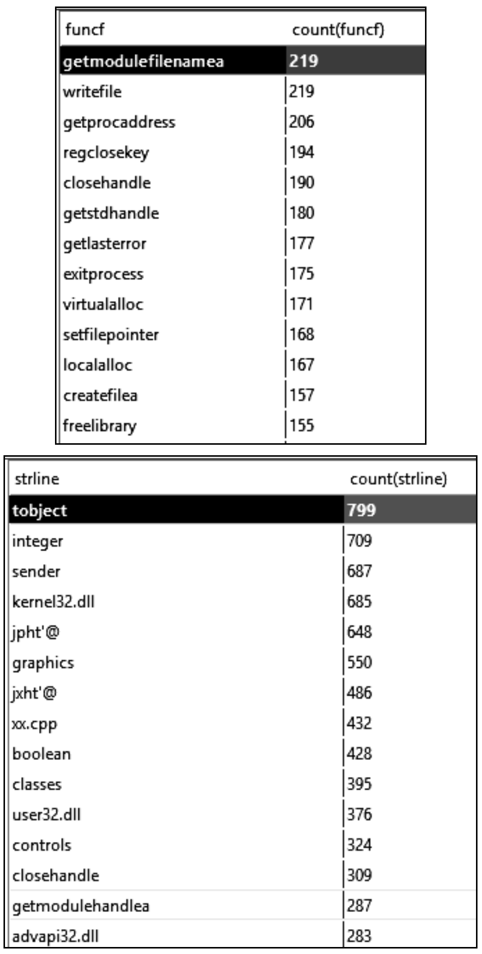
Fig. 5. Fragment of the results of safe software testing

The analysis of the received data of the harmful and secure software, has allowed the allocation of the most frequently meeting functions and strings that are inherent to each family of the considered viruses and the generation of the feature table. It was decided to use 50 features for further analysis. Fig. 6 depicts a table of features that are characteristic for viruses such as Worms.

These features were later used as bit masks for file analysis. As a result of searching for selected features in files, binary vectors of malicious files and safe software were obtained (Fig. 7).

To avoid accidental errors in the transmission of data and to detect intentional changes to the file by attackers, the binary vectors of the malicious software are encoded using one of the MD5, SHA-1, or CRC algorithms.

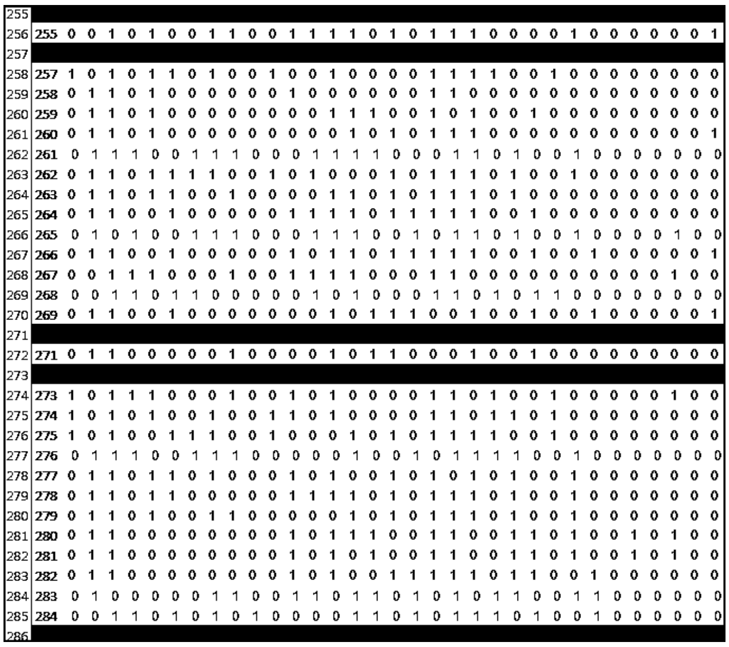
These algorithms are widely used to obtain file signatures. Fig. 8 demonstrates examples of signatures obtained using various methods.



funcf	count(funcf)
getmodulefilenamea	219
writefile	219
getprocaddress	206
regclosekey	194
closehandle	190
getstdhandle	180
getlasterror	177
exitprocess	175
virtualalloc	171
setfilepointer	168
localalloc	167
createfilea	157
freelibrary	155

strline	count(strline)
tobject	799
integer	709
sender	687
kernel32.dll	685
jpht'@	648
graphics	550
jxht'@	486
xx.cpp	432
boolean	428
classes	395
user32.dll	376
controls	324
closehandle	309
getmodulehandlea	287
advapi32.dll	283

Fig. 6. Tables of the most common functions and strings encountered in malicious files such as Worms



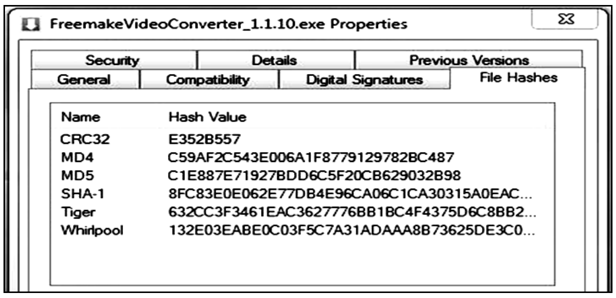
255																																		
256	255	0	0	1	0	1	0	0	1	1	0	0	1	1	1	1	0	1	0	1	1	1	0	0	0	0	1	0	0	0	0	0	0	1
257																																		
258	257	1	0	1	0	1	1	0	1	0	0	1	0	0	1	0	0	0	0	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0
259	258	0	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
260	259	0	1	1	0	1	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0
261	260	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1
262	261	0	1	1	1	0	0	1	1	1	0	0	0	1	1	1	1	0	0	0	1	1	0	1	0	0	1	0	0	0	0	0	0	0
263	262	0	1	1	0	1	1	1	1	0	0	1	0	1	0	0	0	1	0	1	1	1	0	1	0	0	1	0	0	0	0	0	0	0
264	263	0	1	1	0	1	1	0	0	1	0	0	0	0	1	1	0	1	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0
265	264	0	1	1	0	0	1	0	0	0	0	0	1	1	1	1	0	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0
266	265	0	1	0	1	0	0	1	1	1	0	0	0	1	1	1	0	0	1	0	1	1	0	1	0	0	1	0	0	0	0	1	0	0
267	266	0	1	1	0	0	1	0	0	0	0	0	1	0	1	1	0	1	1	1	1	1	0	0	1	0	0	1	0	0	0	0	0	1
268	267	0	0	1	1	1	0	0	0	1	0	0	1	1	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0
269	268	0	0	1	1	0	1	1	0	0	0	0	0	1	0	1	0	0	0	1	1	0	1	0	1	1	0	0	0	0	0	0	0	0
270	269	0	1	1	0	0	1	0	0	0	0	0	0	0	1	0	1	1	1	0	0	1	0	0	1	0	0	1	0	0	0	0	0	1
271																																		
272	271	0	1	1	0	0	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0
273																																		
274	273	1	0	1	1	1	0	0	0	1	0	0	1	0	1	0	0	0	0	1	1	0	1	0	0	1	0	0	0	0	0	1	0	0
275	274	1	0	1	0	1	0	0	1	0	0	1	1	0	1	0	0	0	0	1	1	0	1	1	0	1	0	0	0	0	0	0	0	0
276	275	1	0	1	0	0	1	1	1	0	0	1	0	0	0	1	0	1	0	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0
277	276	0	1	1	1	0	0	1	1	1	0	0	0	0	0	1	0	0	1	0	1	1	1	1	0	0	1	0	0	0	0	0	0	0
278	277	0	1	1	0	1	1	0	1	0	0	0	1	0	1	0	0	1	0	1	0	1	0	1	0	0	1	0	0	0	0	0	0	0
279	278	0	1	1	0	1	1	0	0	0	0	0	1	1	1	1	0	1	0	1	1	1	0	1	0	0	1	0	0	0	0	0	0	0
280	279	0	1	1	0	1	0	0	1	1	0	0	0	0	0	1	0	1	0	1	1	1	0	1	0	0	1	0	0	0	0	0	0	0
281	280	0	1	1	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	1	1	0	0	1	1	0	1	0	0	1	0	1	0	0
282	281	0	1	1	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	1	1	0	0	1	1	0	1	0	0	1	0	1	0	0
283	282	0	1	1	0	0	0	0	0	0	0	0	1	0	1	0	0	1	1	1	1	1	0	1	1	0	0	1	0	0	0	0	0	0
284	283	0	1	0	0	0	0	0	1	1	0	0	1	1	0	1	1	0	1	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0
285	284	0	0	1	1	0	1	0	1	0	1	0	0	0	0	1	1	0	1	0	1	1	1	0	1	0	0	1	0	0	0	0	0	0
286																																		

Fig. 7. An example of a malicious files and secure software scan

The received signatures allowed the formation of a signature database for the examined malicious software.

Further analysis of the software is performed by using a developed code analyzer consisting of an analysis block of the input file's PE structure, a decision-making system, a virus signature base, an output unit.

The decision-making system allows you to set the received signature coefficient of coincidence of the analyzed software with signatures that are stored in the database.



Name	Hash Value
CRC32	E352B557
MD4	C59AF2C543E006A1F8779129782BC487
MD5	C1E887E71927BDD6C5F20CB629032B98
SHA-1	8FC83E0E062E77DB4E96CA06C1CA30315A0EAC...
Tiger	632CC3F3461EAC3627776BB1BC4F4375D6C8BB2...
Whirlpool	132E03EABE0C03F5C7A31ADAAA8B73625DE3C0...

Fig. 8. Examples of file signatures

The results of testing the developed parser showed the possibility of using it to detect modified malware at 92% coincidence with the signatures that are stored in the database. With a decrease in the coefficient of coincidence, false positives appear which require additional investigation of the analyzed file, for example, by introducing a PRL block based on a nerve network.

Conclusions

In this paper, we propose a software model of static file detection, based on the analysis of the PE file structure. 3500 malicious files (such as Worms, Backdor, Trojan) and safe software have been scanned; sections of file structure import and export have been analyzed. Features in the form of strings and API functions inherent in these families of viruses have been selected; virus signatures have been generated and stored in the signature database. Using the developed software model, a static parser of malicious files has been tested to detect modifications of malicious software.

The test results revealed the possibility of using the developed automatic static parser of malicious files in the general system of anti-virus data protection. At the same time, the coefficient of coincidence of the received signatures of files with the masks template is high enough, and its reduction leads to false positives. This disadvantage can be eliminated by introducing into the decision-making system an additional analysis block, for example, based on a nerve network.

REFERENCES

1. Polugodovoy otchet po IB ot Cisco [Semi-annual report on information security from Cisco], available at: http://www.securitylab.ru/blog/personal/ Informacionnaya_bezopasnost_v_detalyah/316275.php (last accessed February 28, 2017).
2. Shelukhin, O.I., Sakalema, D.Zh. and Filinov, A.S. (2013), Obnaruzhenie vtorzheniy v kompyuternyie seti [Intrusion Detection into Computer Networks], Moskva : Hot line-Telecom, 220 p.
3. Semenov. S.G., Davydov, V.V., and Gavrilenko, S.Yu (2014), Zaschita dannyih v kompyuterizirovannyih upravlyayuschih sistemah (monografiya) [Data Protection in Computer-Aided Control Systems (monograph)] , "LAP LAMBERT ACADEMIC PUBLISHING" Germany, 236 p.
4. Igray, kak "Laboratoriya Kasperskogo" [Play as "Kaspersky Lab"], available at: http://www.kaspersky.ru/about/news/product/2017/kompaniya-otkryvayet-dostup-k-svoyey-baze-znaniy-o-kiberugrozakh-v-ramkakh-novogo-biznes-servisa (last accessed February 28, 2017).

5. Lukatsky, A.V. (2001), *Obnaruzhenie atak* [Attack Detection], St. Petersburg : VHV-Petersburg, 624 p.
6. Kaspersky, K. (2006), *Zapiski issledovatelya kompyuternyih virusov* [Notes of a researcher of computer viruses], St. Petersburg: Peter, 316 p.
7. Goshko, S.V. (2009) *Tehnologii borbyi s kompyuternyimi virusam*i [Technologies to combat computer viruses], Moscow: Solon-Press, 352 p.
8. Semenov, S., Gavrilenko, S. and Chelak V. (2016), "Developing parametrical criterion for registering abnormal behavior in computer and telecommunication systems on the basis of economic test", *Actual problems of economics*, Kiev, Vol 4 (178), pp. 451–459.
9. Tolstikhin I.O. (2009), *Razrabotka metodov klassifikatsii zlovrednyih ispolnyaemyih faylov* [Development of classification methods for malicious executable files], available at: http://www.machinelearning.ru/wiki/images/ 5/58/Tolst09techrep.pdf (last accessed February 28, 2017).
10. Ero Carrera (2007), *Win32 Static Analysis in Python*, available at: http://2006.recon.cx/en/f/lightning-ecarrera-win32-static-analysis-in-python.pdf (last accessed February 28, 2017).
11. Sikorski, M. (2012) A. *Honig Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software*: San Francisco, 802 p.
12. AntivIrusnI tehnologIyi: v poshukah panatseyi [Antivirus technologies: in search of a panacea], available at: http://zillya.ua/antivirusni-tekhnologi%D1%97-v-poshukakh-panatse%D1%97 (last accessed February 28, 2017)
13. John Snow (2016), S*ozdaem PE-virus №1* [Create PE-virus No.1], available at: https://xakep.ru/2007/04/23/37880/ (last accessed February 28, 2017).
14. Obnaruzhenie, osnovannoe na signaturah [Signature-based detection], available at: http://mind-control.wikia.com/wiki (last accessed February 28, 2017).
15. PE Detective, available at: http://ntcore.com/pedetective.php (last accessed February 28, 2017).
16. Antivirusnyie dvizhki [Antivirus engines], available at: https://fcenter.ru/online/ softarticles/utilities/12214 (last accessed February 28, 2017).

Надійшла (received) 31.03.2017
Прийнята до друку (accepted for publication) 13.06.2017

Розробка методу і програмної моделі статичного аналізатора шкідливих файлів

С. Ю. Гавриленко, Д. М. Саєнко

Предметом дослідження в даній статті є методи аналізу шкідливого програмного забезпечення. **Мета** статті полягає в підвищенні безпеки функціонування комп'ютерних систем (КС) і захисту їх від впливу комп'ютерних вірусів. **Завдання:** дослідження сучасних засобів антивірусного захисту програмного забезпечення; аналіз методів формування сигнатури файлів; розробка програмної моделі статичного детектування файлів, що базується на аналізі PE-структури; формування таблиць ознак, притаманних родин вірусів типу Worms, Backdor, Trojan; отримання довічних сигнатур шкідливого і безпечного програмного забезпечення. Використовуваними **методами** є: аналіз коду в Hex-файлі, алгоритми хешування файлів. Отримані наступні **результати**. Проаналізовано PE-структуру файлу, обрані секції для подальшого аналізу. Розроблена програмна модель статичного детектування файлів і виконано аналіз безпечних і шкідливих файлів. Обрані ознаки у вигляді рядків і API функцій, сформована бітова маска для подальшого аналізу файлів. Виконано сканування 3500 файлів шкідливого і безпечного програмного забезпечення, проведено їх аналіз. Сигнатури кожного шкідливого файлу закодовані і збережені в базі сигнатур, За допомогою розробленої програмної моделі виконано дослідження можливості виявлення модифікацій шкідливого програмного забезпечення. **Висновок**. Розроблено метод і програму модель статичного детектування шкідливих файлів, що дозволяє отримати набір ознак файлу в автоматичному режимі і зробити висновок про шкідливість файлу.

Ключові слова: шкідливе програмне забезпечення, сигнатура, Python, портативний запуск, шкідливий додаток, API-функції, шкідливі файли.

Разработка метода и программной модели статического анализатора вредоносных файлов

С. Ю. Гавриленко, Д. Н. Саенко

Предметом исследования в данной статье являются методы анализа вредоносного программного обеспечения. **Цель** – повышении безопасности функционирования компьютерных систем (КС) и защита их от воздействия компьютерных вирусов. **Задачи**: исследование современных средств антивирусной защиты программного обеспечения; анализ методов формирования сигнатуры файлов; разработка программной модели статического детектирования файлов, базирующаяся на анализе PE-структуры; формирование таблиц признаков, присущих семействам вирусов типа Worms, Backdor, Trojan; получение двоичных сигнатур вредоносного и безопасного программного обеспечения. Используемыми **методами** являются: анализ кода в Hex-файле, алгоритмы хеширования файлов. Получены следующие **результаты**. Проанализирована PE-структура файла, выбраны секции для последующего анализа. Разработана программная модель статического детектирования файлов и выполнен анализ безопасных и вредоносных файлов. Выбраны признаки в виде строк и API функций, сформирована битовая маска для дальнейшего анализа файлов. Выполнено сканирование 3500 файлов вредоносного и безопасного программного обеспечения, проведен их анализ. Сигнатуры каждого вредоносного файла закодированы и сохранены в базе сигнатур, С помощью разработанной программной модели выполнено исследование возможности обнаружения модификаций вредоносного программного обеспечения. **Выводы**. Разработан метод и программная модель статического детектирования вредоносных файлов, позволяющая получить набор признаков файла в автоматическом режиме и сделать вывод о вредоносности файла.

Ключевые слова: вредоносное программное обеспечение, сигнатура, Python, портативний запуск, вредоносное приложение, API-функции, вредоносные файлы.