UDC – 004.725:004.852

DATA PREPARATION MODEL FOR TRAINING NEURAL NETWORKS

Artak A. Khemchyan
National Polytechnic University of Armenia
105, Teryan St., 0009, Yerevan
a.khemchyan@polytechnic.am
ORCID iD: 0009-0008-3271-1903
Republic of Armenia

Timur V. Jamgharyan
National Polytechnic University of Armenia
105, Teryan St., 0009, Yerevan
t.jamgharyan@yandex.ru
ORCID iD: 0000-0002-9661-1468
Republic of Armenia

https://doi.org/10.56243/18294898-2023.4-28

Abstract

The paper presents the results a research of changes in the state of the operating environment when it is damaged by malware. The Windows operating system of various versions and builds was selected as a test environment. The research was carried out using polymorphic malware *abc, cheeba, december_3, stasi, otario, dm, v-sign, tequila, flip*. The research was conducted to obtain the value of operating system signatures for subsequent training of neural networks. Context triggered piecewise hashing and performance testing methods were used for research the assessment of changes in the state of the operating environment when it is damaged by malware. Simulation of the developed method was carried out in the Hyper-V virtual environment.

Keywords: polymorphic malware, software testing method, data reliability, context triggered piecewise hashing, Parrot OS, svchost.

Introduction

The active development of machine learning (ML) technology has provided attackers with new tools for developing malware. In particular, if an attack on the network infrastructure (NI) will sooner or later be detected, then an attack against the operating system (OS) threatens the *reliability of the processed and stored data*[1]. The danger lies in the fact that the use of proprietary operating systems with closed source code makes it impossible to accurately assess the integrity of the operating system during operation, since it becomes

[1] Data reliability is the property of the processed data not to have hidden errors [1].

unclear whether the operating system is affected by malware or whether an undeclared feature introduced by the manufacturer itself is activated. It is possible to evaluate only the image of the official distribution. In proprietary operating systems, analyzing the use of functions/libraries and other components is quite difficult due to the lack of source code. Based on the above, an urgent task is to study the behavior of the OS before and after being damaged by malware. Among the many types of malware, polymorphic[2] malware plays a special role. This research examines the behavior of the Windows OS of various versions and builds when it is infected with polymorphic malware. Since malware is also built on the basis of a specific code base, various software testing methods are also applicable to it (*configuration testing method, regression testing method, gray, white, black box method, functional testing method, performance evaluation method* [2,3]). It becomes possible to infect a system with previously known malware and evaluate its behavioral model using specified parameters and calculate a signature. The solution to this problem allows us to solve the inverse problem: to assess the degree of infection of the OS by its «signature cast». Various researchers are trying to solve the problem of detecting malware activity in an OS using different methods, but each of them considers a specific OS [4,5,6,7]. The novelty of the research lies in the use of software *performance testing* methods and *context triggered piecewise hashing*[3] to assess changes in the state of the operating environment when it is infected with malware.

Conflict Setting

It is necessary to obtain an OS signature value when exposed to malware with specified parameters.

Discussion

Various polymorphic malware, the source code and the modification algorithm of which are known, are gradually being introduced into the Windows operating system of various versions and builds (with a known, verified hash value of the unaffected version).

The workload of the *svchost* process is calculated and the context triggered piecewise hashing OS hash value is calculated. The choice of the svchost process to evaluate the OS state is due to the fact that *svchos*[4]*t* is the main process when activating service processes and loading dynamic libraries. All other processes are children of *svchost*. Thus, it is possible to have a system not affected by malware, record the value of svchost and compare it with the value of the affected system at different points in time. Creating a signature base of the state of the operating environment allows you to evaluate its current state and assess the degree of infection by malware. The Windows operating systems (versions, build numbers) analyzed are presented in tab 1.

[2] Polymorphic malware is software that is characterized by the following behavior: encryption, self-propagation and modification of one and/or several components of the source code.
[3] Context triggered piecewise hashing (CTPH) is a method for computing piecewise hashes from input data [8].
[4] svchost.exe in the Microsoft Windows family of operating systems is the main process for services loaded from dynamic libraries [9].

Table 1

Windows operating systems (versions, build numbers) analyzed

Windows 7	Windows 10		Windows 10 IoT	Windows Server 2016 (version 1607/1709/1803)
version	version	build	build	build
7077	1809	17763	16299-1	10.0.09841
7100	1903	18362	16299-2	10.0.10074
7227	1909	18363	18362	10.0.10537
7228	2004	19041	19041	10.0.10586
7270	20H2	19042	19043	10.0.14300
7271	21H1	19043	18363	10.0.14393

Experimental procedures

On the Dell Power Edge T-330 server, the Hyper-V role is installed in the Windows Server 2016 Standart operating system environment. A SDN (Software Defined Networking) has been deployed in which Parrot OS with the Metasploit framework installed and various versions (builds) of Windows OS downloaded from the official website are installed [10].

Table 2

Windows operating systems (versions, build numbers) analyzed

Build number/ OS version	Unaffected OS image hash value	Hash value of the image of the affected OS malware *abc, cheeba, december_3* (CTPH value 64 bytes)	Hash value of the image of the affected OS malware *abc, cheeba, december_3* (CTPH value 128 bytes)	Hash value of the image of the affected OS malware *abc, cheeba, december_3* (CTPH value 256 bytes)
Windows 10				
1909/ 18363	6fba99620cf84e6933b140027ce9fbbeed51bf23	31e5aafa83e17c999142b7aa270d6a55f777afcd	cf63fe5a9ce046eb7bb068e95d5ee79e9ebd696b	72a096e3d803d7585b2e9be0ce5debfe5a5c1552
20H2/ 19042	b70a55869178c6927ec29171b6afd563686e8efe	3851c4417b41a48877f8644d60a8d7d528dd5c92	358879ccd1e04b5b3149386dd80d588abd7e1b6d	4a99c6be6113f3318f1040c7fb1bd2d39b37b55d
Windows 10 IoT				
18362 19043	632667547e7cd3e0466547863e1207a8c0c0c549	11fbf8b0fae93c46ae6fa191bc67daf114a8b573	909eb9d63b44be52c318b10f6d538c552a7133a3	d21c11f950477838d9b5544d55a4192b9e97371d
Windows Server 2016				
10.0.10537	f7f2c3285303b9ab412da2d7e3e453488b40f6ab	50996e82617dcef6073cf328e3dfef7ccd3ba20d	fd8f08f0541cf41438d6e400f0682ccb457fe663	f1d670bc47ef19755b6c57bd0f6180793483a2bb
10.0.14300	346b8b56fc47599ae393a5de4afb373a05216c55	ad04ced83163588f0831304c632084b6585f8ddd	f209bcf375dfecce93c545a2208273d57659bf90	325cdc77897bcf7ce0c10276650b56ac777a50d3

Within virtual machines, various combinations of the number of processors and RAM are configured. The primary assessment was the CPU load without damaging the OS by

malware. Load estimation was done by measuring the value of the svchost process. After each measurement, the OS CTPH value was calculated. The second stage was to gradually introduce malware *abc, cheeba, december_3, stasi, otario, dm, v-sign, tequila, flip* into the OS and measure the state of the *svchost* process and the OS CTPH value. The CTPH values for Windows operating systems affected by malware are presented in tab. 2.

The malware was introduced using the Metasploit framework. The assessment of changes in the state of polymorphic software was carried out by comparison with the source code based on the method proposed in [11]. The svchost process load was measured using process explorer software from the Sysinternals (Winternals) software package [12]. The assessment of changes in the state of the operating system was carried out using the piecewise context hashing method with a variable hashing step size using *ssdeep* software [13]. In all cases, testing was carried out with anti-malware software disabled and the OS updated to the latest state. No additional application software that could affect the results of the study was installed.

Research Results

Fig. 1-4 shows the results of visualization of the recycling of the CPU of a virtual machine with Windows 7, Windows 10, Windows 10 IoT, Windows Server 2016 installed when it is damaged by malware *abc, cheeba, december_3, stasi, otario, dm, v-sign , tequila, flip*.

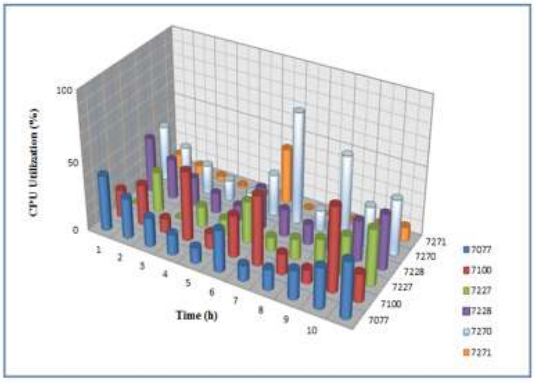

Fig. 1 Visualization of CPU utilization running the Windows 7 operating system, if it is infected with malware

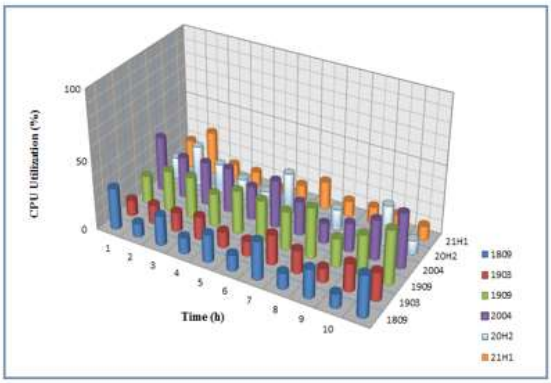

Fig. 2 Visualization of CPU utilization running the Windows 10 operating system, if it is infected with malware

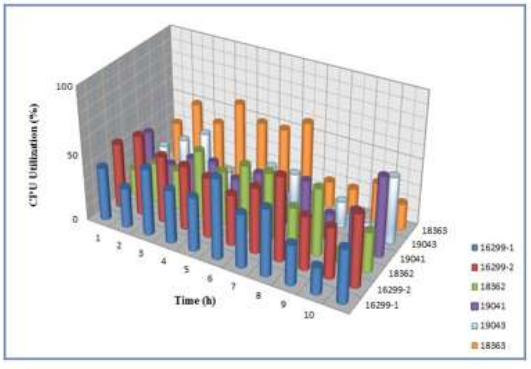

Fig. 3 Visualization of CPU utilization running the Windows 10 IoT operating system, if it is infected with malware

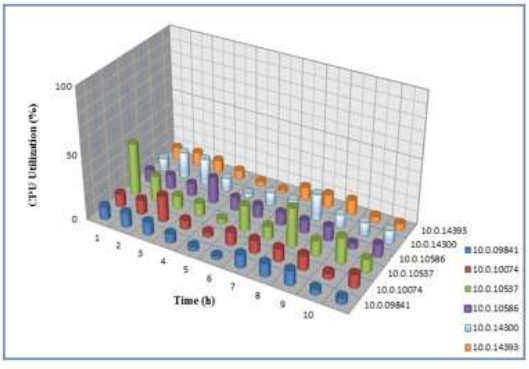

Fig. 4 Visualization of CPU utilization running the Windows Server 2016 operating system, if it is infected with malware

The virtual machine is configured in the following configuration: 4 cores with a processor frequency of 3100 Mhz and a RAM capacity of 32Gb.

When the operating system is infected with malware, additional activation of svchost child processes occurs. Increasing the amount of malware embedded in the OS increases the number of child svchost processes, correspondingly increasing processor utilization, which as a result can lead to a hidden attack on availability. In all types used, polymorphic software affected the *hal.dll*[5] and *datachannel.dll*[6] libraries. In Windows 7, both libraries are affected, increasing the level of processor utilization; in Windows 10, of the system libraries, only the *datachannel.dll* library is affected, which ultimately reduces processor utilization, but makes the system more affected since malicious code is transferred between different processes (programs)) by modifying your code within a single OS. It is also possible for malware to be transferred undetected between different hosts. Knowing the value of the hash function under various OS states allows you to build a dataset for training neural networks integrated with the SIEM (Security information and event management) system that monitors hosts with Windows OS installed.

Conclusion

The paper discusses a model for determining whether an OS is affected by polymorphic malware, built on the methods of CTPH and software performance testing. The analysis of CPU utilization after the defeat of the operating system was determined by the number of svchost processes and the degree of computing resource they consumed. It was determined that in all cases the polymorphic malware used primarily affected processes associated with the use of the dynamic libraries *hal.dll* and *datachannel.dll*. The most vulnerable operating systems were Windows 7 (builds 7100, 7227, 7228), Windows 10 IoT (builds 18362, 19041, 19043. The least vulnerable OS is Windows Server 2016. In Windows Server 2016 build 10.0.14393, the *datachannel.dll* library was blocked if the Hyper-V role was activated and a Windows OS was running in a virtual environment. Based on the research, the values of CTPH of unaffected OS and affected OS were cataloged at CTPH step values of 64, 128, 256, 512 bytes. The research all results are presented in [14].

References

1. National standard of the Russian Federation, «Quality of service information», GOST R-51170-98, (2020) // 12, Moscow, Standardinform.
2. Myers G., Badgett T., Sandler K., The Art of Software Testing, 3rd Edition. - M.: Dialectics, 2012. - 272 p.
3. Sinitsyn S., Nalyutin N., Software verification. - M.: BINOM, 2008. - 368 p.
4. Abusnaina A., et al, «Burning the Adversarial Bridges: Robust Windows Malware Detection Against Binary-level Mutation». https://doi.org/10.48550/arXiv.2310.03285

[5] hal.dll (hardware abstraction layer) is a library responsible for the interaction of the software and hardware parts of the computer.
[6] datachannel.dll - a library responsible for ensuring data transfer between various programs and components.

5. Anand S., et al, «MALITE:Lighweight Malware detection and Classification for Constrained Devices», https://doi.org/10.48550/arXiv.2309.03294
6. Islam N., Shin S., «Review of Deep Learning – based Malware Detection for Android and Windows System», https://doi.org/10.48550/arXiv.2307.01494
7. Yousuf M.I, et al, «Multi-feature dataset for Windows PE Malware Classification». https://arxiv.org/abs/2210.16285
8. Kornblum J., (2006), «Identifying Almost Identical Files using Context Triggered Piecewise Hashing. Digital Investigation» 3, 91-97. Digital Investigation. 3. 91-97. 10.1016/j.diin.2006.06.015.
9. Download page and description of the svchost process., https://learn.microsoft.com/en-us/windows/application-management/svchost-service-refactoring «Changes to Service Host grouping in Windows 10», Microsoft. 2021-08-27. Retrieved 2021-01-10.
the resource is available on 20.12.2023.
10. Official download page for various versions of the Windows operating system, https://www.microsoft.com/ru-ru/software-download , the resource is available on 20.12.2023.
11. Jamgharyan T., (2022). «Research of Obfuscated Malware with a Capsule Neural Network». *Mathematical Problems of Computer Science*, *58*, 67–83. https://doi.org/10.51408/1963-0094
12. Official operating system testing tools download page Windows, Sysinternals, https://learn.microsoft.com/en-us/sysinternals/ , the resource is available on 20.12.2023.
13. Ssdeep software download page. https://ssdeep-project.github.io/ssdeep/index.html , the resource is available on 20.12.2023.
14. All research results.https://github.com/T-JN , the resource is available on 20.12.2023.

References

1. Национальный стандарт Российской Федерации, «Качество служебной информации», ГОСТ Р-51170-98, (2020)// 12, Москва,Стандартинформ.
2. Майерс Г., Баджетт Т., Сандлер К.. Искусство тестирования программ, 3-е издание, The Art of Software Testing, 3rd Edition. — М.: «Диалектика», 2012. - 272 с.
3. Синицын С., Налютин Н., Верификация программного обеспечения. - М.: БИНОМ, 2008. — 368 с.
4. Abusnaina A. et al, «Burning the Adversarial Bridges: Robust Windows Malware Detection Against Binary-level Mutation». https://doi.org/10.48550/arXiv.2310.03285
5. Anand S., et al, «MALITE: Lighweight Malware detection and Classification for Constrained Devices», https://doi.org/10.48550/arXiv.2309.03294
6. Islam N., Shin S., «Review of Deep Learning –based Malware Detection for Android and Windows System». https://doi.org/10.48550/arXiv.2307.01494

7. .Yousuf M.I, et al, «Multi-feature dataset for Windows PE Malware Classification». https://arxiv.org/abs/2210.16285
8. Kornblum J., (2006), Identifying Almost Identical Files using Context Triggered Piecewise Hashing. Digital Investigation 3, 91-97. Digital Investigation. 3. 91-97. 10.1016/j.diin.2006.06.015.
9. Страница загрузки и описания процесса svchost. https://learn.microsoft.com/en-us/windows/application-management/svchost-service-refactoring «Changes to Service Host grouping in Windows 10», Microsoft. 2021-08-27. Retrieved 2021-01-10, the resource is available on 20.12.2023.
10. Официальная страница загрузки различных версий операционной системы Windows, https://www.microsoft.com/ru-ru/software-download the resource is available on 20.12.2023.
11. Jamgharyan T., (2022). «Research of Obfuscated Malware with a Capsule Neural Network». *Mathematical Problems of Computer Science*, *58*, 67–83. https://doi.org/10.51408/1963-0094
12. Официальная страница загрузки инструментов тестирования операционной системы Windows, Sysinternals. https://learn.microsoft.com/en-us/sysinternals/,the resource is available on 20.12.2023.
13. Официальная страница загрузки ПО ssdeep. https://ssdeep-project.github.io/ssdeep/index.html , the resource is available on 20.12.2023.
14. Полные результаты исследования. https://github.com/T-JN , the resource is available on 20.12.2023.

ՆԵՅՐՈՆԱՅԻՆ ՑԱՆՑԵՐԻ ՈՒՍՈՒՑՄԱՆ ՀԱՄԱՐ ՏՎՅԱԼՆԵՐԻ ՆԱԽԱՊԱՏՐԱՍՏՄԱՆ ՄՈԴԵԼ

Խեմչյան Ա.Ա., **Ջամղարյան Թ.Վ.**
Հայաստանի ազգային պոլիտեխնիկական համալսարան

Ներկայացված են օպերացիոն միջավայրի վիճակի փոփոխությունների ուսումնասիրության արդյունքները, վնասաբեր ծրագրային ապահովումով գրոհի դեպքում: Որպես թեստային օպերացիոն միջավայր ընտրվել է տարբեր տարբերակների և կառուցվածքների Windows օպերացիոն համակարգը: Հետազոտությունն իրականացվել է վնասակար պոլիմորֆ ծրագրային ապահովման միջոցով՝ *abc, cheeba, december_3, stasi, otario, dm, v-sign, tequila, flip*: Հետազոտությունում ստացված օպերացիոն համակարգի սիգնատուրաների ցուցանիշները հետագայում կիրառվել են նեյրոնային ցանցերի ուսումնառության համար: Համատեքստի մասնակի հեշավորման և արտադրողականության փորձարկման մեթոդները կիրառվել են որպես գործող օպերացիոն միջավայրի վիճակի փոփոխությունների գնահատման մեթոդներ, երբ այն

վնասված է վնասաբեր ծրագրային ապահովումով: Մշակված մեթոդի մոդելավորումն իրականացվել է Hyper-V վիրտուալ միջավայրում:

Բանալի բառեր. *պոլիմորֆ ծրագրային ապահովում, ծրագրային ապահովման փորձարկման մեթոդ, փվյալների հուսալիություն, համապեքսփր հատվածային հեշինգ, Parrot OS, svchost:*

МОДЕЛЬ ПОДГОТОВКИ ДАННЫХ ДЛЯ ОБУЧЕНИЯ НЕЙРОННЫХ СЕТЕЙ

Хемчян А.А., Джамгарян Т.В.
Национальный политехнический университет Армении

Представлены результаты исследования изменения состояния операционной среды при поражении ее вредоносным программным обеспечением. В качестве тестовой среды, выбрана операционная система Windows различных версий и сборок. Исследование проводилось с применением вредоносного полиморфного программного обеспечения *abc, cheeba, december_3, stasi, otario, dm, v-sign, tequila, flip.* Исследование проводилось с целью получения значения сигнатур операционной системы для последующего обучения нейронных сетей. В качестве методов исследования оценки изменения состояния операционной среды при поражении ее вредоносным ПО применялись *методы кусочно-контексного хэширования и тестирования производительности.* Моделирование разработанного метода проведено в виртуальной среде Hyper-V.

Ключевые слова: полиморфное ПО, метод тестирования ПО, достоверность данных, кусочно-контекстное хэширование, Parrot OS, svchost.

Submitted on 06.11.2023
Sent for review on 13.11.2023
Guaranteed for printing on 25.12.2023