VIRTUAL SYSTEM OBFUSCATION

Michal HULIČ, Martin CHOVANEC, Viliam KORBA
Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics,
Technical University of Košice, Letná 9, 042 00 Košice, Slovak Republic, tel. +421 55 602 4220,
E-mail: michal.hulic@tuke.sk, martin.chovanec@tuke.sk, viliam.korba@student.tuke.sk

ABSTRACT

This publication deals with obfuscating of virtual systems against malware. It describes virtual systems, malicious code, weaknesses of virtual systems and methods of obfuscating. The addresses at the type of virtual system, use in practice, definition of malware and methods of avoiding systems designed to detect malicious software. Findings are applied in the design and subsequently in the implementation of the tool. The tool uses wear and tear artefacts to obfuscate virtual systems which are detecting malicious software. Designed and implemented software section of this publication is launched in virtual system with operation system Windows.

Keywords: *Virtual systems obfuscation, Malware, Wear and tear artefacts, Virtual systems, Detecting virtual systems.*

1. INTRODUCTION

The problem of obfuscation virtual systems is one part of a fight against malware attacks. The safest and most common analysis environment for Malicious software is a virtual environment where malicious sensitive software or system resources are not affected by the software. In this case, the attackers are so sophisticated that they can analyze the environment and, on that principle, evaluate whether they trigger their malicious behavior or malicious software leaves for the sleeping phase. In the sleeping phase malicious software is not identifiable because it does not execute malicious operations and emulates harmless software. This publication emulates the operating system in a virtual environment Windows 10 from vendor Microsoft so that it is from the attacker's point of view, the same as the real system user. The work is focused on parts of the operating system that are analyzed by malicious software. It's been investigated and identified under what conditions or events changes their status. Then the options like that were found and adjusted to the state of the desired parts and implemented them into the tool. On the basis of this information there was a need to design an obfuscation tool. There must be selected the right strategy and focus on the most essential features. Finally, the implementation was verified and determined effectiveness of the tool. The aim of the tool is to reduce the similarity of the environment to environments that analyze malicious software.

2. VIRTUAL SYSTEMS

Virtualization is an activity of involving a creation of virtual environment on one machine and partition its computing power and storage for more concurrent running operating systems (clients). Also, the host computer hardware appears to be virtual, what allows us to run applications that are designed for another architecture or test purposes, since the OS does not direct access to hardware. Hypervisor is hiding from physical hardware available to users instead virtual hardware. This software runs on a physical computer, manages virtualization and provides services to clients. There are five kinds of virtualization based on what it is. Types of virtualization: Hardware virtualization, data virtualization, desktop virtualization, operating virtualization and system virtualization of network functions.

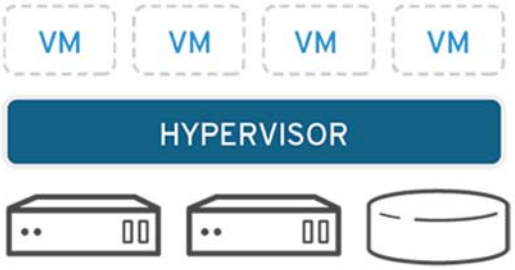

Fig. 1 Virtualisation [6]

Operating system virtualization creates virtualized operating system-level hardware to create multiple isolated virtualized instances that run on one system. This process is performed without use of hypervisor. Operating system virtualization has the same guest OS as host. The OS virtualization uses the host OS like base of all independent virtual machines in the system. Operating system virtualization does not need driver emulation. This leads to better performance and the ability to run multiple virtualizations simultaneously.

2.1. Hardware virtualization

Also known as server virtualization. Servers are designed to perform intensive tasks which needs to be performed in a short amount of time (fractions of milliseconds). In addition, servers are often required to do tasks of different kinds for which are required different environment. Virtualization helps solve this situation for servers. One machine divides the parts as needed and specific tasks are available for each specific task environment [1]. Because in this kind of virtualization is a hypervisor installed directly on computer hardware, and others operating systems are installed later on. This procedure is also known as bare-metal virtualization. Based on this the hypervisor is considered to be a host operation system with its own rights, although servers mostly use a virtual machine with an operating system.

3. MALWARE

Malicious software, also known as malware, is a term that consists of two words: malicious, meaning dangerous and the

word software. This software is embedded in the system to compromise confidentiality, the integrity or availability of software resources. While in security by the term threat is thought possible security breaches which may arise if: there are circumstances, possibilities, abilities of action or events that can break safety and cause damage. This is a danger that can exploit vulnerability system. Vulnerability is a mind error or weakness system design of implementation or operation which can be exploited to compromise system security. The security confidentiality is divided into data confidentiality and personal information. Data confidentiality is private and confidential information shall not be accessible to unauthorized persons. The confidentiality of personal data means that the attackers who collect and store user information for the purpose blackmail or deceiving users will not be accessible.

We also divide the integrity of the system into two groups: data integrity and system integrity. Data integrity means that information and programs are changed only in a specific and justified manner. System integrity again ensures that the system performs the intended function respectively. Operation without interruption any intentional or involuntary unauthorized handling. The availability of the system ensures that the systems work accurately, and the service is not denied by authorized users. Mostly, this software is embedded and hidden where its intention is also to remain secret [2].

4. DETECTING THE ENVIRONMENT BY ATTACKING SOFTWARE

Malicious software recognizes the environment it is in to find out whether the environment is a sandbox system or a virtual environment. If it's the sandbox system where it's being operated, then there are two options malicious system behavior [3]. A malicious system interrupts execution and its processes are destroyed. This behavior is malicious because of the tools on they will not be available to analyze the malicious system no behavior to analyze and chances increase evaluation of the system as harmless. The malicious system will start harmless operations. In this case, it is possible to detect a malicious system. To detect the presence of a virtual machine or several technologies are used in the sandbox system. We need to know these technologies if we want to cover the presence of the sandbox system. If we want to hide the presence of the sandbox system is necessary to treat all the different features of the sandbox system from real user system [4].

Recognition technologies:

- The presence of the Hypervisor
- specific hardware features
- using specific knowledge about intermediaries
- use of sandboxing technologies
- using artificial environment
- based on time
- user interaction
- system interaction
- obfuscating of internal data
- based on wear and tear artifacts

If we hide these features from the sandbox system, it is theoretically difficult to discern. If the system will have all the features of both the real user system and so, on malicious software will take longer to analyze the system and this will give us time for the malicious tool system.

4.1. Presence of hypervisor

In the past, when virtual machines did not have hardware support and could be recognized by technical artifacts that could be found on certain approaches. For example, through a virtual machine from VMWare it was 0x5658. The second option was disclosure generic hypervisor's artifacts. After arrival of a hardware support, these techniques are not used because these approaches are no longer visible [5].

Nowadays, a useful technique is the revelation hypervisor's implementation artifacts. That is meant to get the MAC address, device identification number or control unit identification number. Another way of recognizing the presence of the hypervisor is distinguishing values registers. For example, register "HKEY_LOCAL_MACHIN\ HARDWARE\DESCRIPTION\System" is associated with virtual machine implementations.

4.2. Recognize a virtual machine based on specific hardware properties

One of the Techniques of Detecting a Virtual Machine based on specific hardware features is the technique called Red Pill. The basic part of the code is SIDT (save instruction) interrupt descriptor tables Store interrupt descriptor table) instruction. This instruction stores content of register to the interruption descriptor table (Interruption Descriptor Table Register - IDTR) to target a surgeon who is actually a memory location. SIDT the instruction is accessed by non-privileged modes, returning sensitive registry content used by the operating system.

Because of the processor has only one IDTR register containing the descriptor table interruption, so in the case of two running operational systems, which is a case of virtualization, is a collision. One of the operating systems is the host and another one is the guest (or real system and virtual machine). The host operating system must allocate the IDTR to the guest to avoid collisions between host and guest registers. The virtual machine cannot know if it is or when it is the SIDT instruction is executed because it has no privileges to do so. This means that the process gets a new allocated address. The Red Pill function compares whether the original IDT address was not modified and returns 1 as a result if is in the virtual machine and the value 0 if It is not. This strategy is effective on hardware, based on Intel processor architecture [7].

4.3. Utilization of specific knowledge of the intermediary

Virtual environments have specific features. The creators of malicious systems study these properties and they try to detect them through malicious software. These pages collect data about the most commonly used virtual environments and their specific characteristics; and they implement them in their analysis tools. One of these properties is the ecosystem it represents the mechanisms necessary to return the analytical environment to its original clean state. E.g. freeze system. Or creating communication with different approaches such specific network environment. Sandbox can also have specific files, processes, specific file structure, specific drivers or specific username [8].

4.4. Using sandboxing technologies

Most sandbox systems use technology called a hook. Hooks are specifically meant to edit a code and data to intercept communications processes, drivers, and operating system. And that's how it makes them recognizable.

- Usually check certain instructions or indicator
- Verify system integrity

Some sandboxes utilize the emulsions they have side effects and it is difficult to distinguish them from the native operating system. These systems have, for example different instruction semantics, cache-based attacks memory. The emulation gaps can be detected by invoking unclear instructions of the control unit.

4.5. Based on artificial environment

Sandboxes are special systems for analysis of a malicious system, and that’s why they are not entirely identical to real operating systems. That means they are recognizable by malicious software. Differences between sandbox and real operating system are:

- Hardware features: unusual low-resolution screen, no USB drivers, small memory sizes.
- Software features: no email account, untypical software stack.

4.6. Time-based recognition of the virtual environment

In case of the harmful system prolongs the sleeping phase after penetration into the virtual environment respectively. sandbox, most often it successfully leaves the sandbox system before activation phase. In cases where the software is malicious programmed to execute execution in a certain date and at a certain time. In this case, it also leaves sandbox system before running. Malicious software can contain malicious code that performs unnecessary cycles to the central processing unit during the execution of its real intention until the sandbox testing ends [9].

Monitoring the behavior of applications with sandbox analysis is more time consuming to detect a malicious system process as when the application is launched without analysis. this feature can detect and evaluate malicious software that is in the sandbox. Sandboxes try this to fix the problem so that it is time but harmful the system can include an external time source bypassing and covering up the sandbox analysis [10].

4.7. Virtual environment recognition based on user interaction

There are cases where a malicious system monitors user interaction. User is interactive with graphical interface in several possible ways:

- Scrolling through a document
- Pressing the mouse buttons
- Mouse movement
- Keyboard keystroke

A malicious system can occur in one document in which it can wait to browse through a certain party. Most often this technique was implemented in Microsoft Office software. After fulfillment conditions, the malicious software activated and fired the bomb. Another implementation of this technique is motion tracking mouse from entering malicious system into the environment. If mouse is at all times located in the middle of the operating screen so it is likely that the system is designed for analysis of malicious systems.

4.8. Obfuscation of internal data

Malicious software can encrypt application calls programming interface and therefore sandbox is unable to read these calls and cannot evaluate the call software. The same was for Trojan Dridex horse. Furthermore, a malicious system can change the domain names and IP addresses by which it intends to hide practices attackers who are trying to trick the user for the purpose obtaining its sensitive data based on which they could get money. In this way the attacker covers up its actual address. The attackers are not subsequently registered in the blacklist on pages that use security systems.

4.9. Virtual environment recognition based on wear and tear artifacts

In this technology, the hypervisor is not recognized instead virtual environment is. Sandbox itself is recognized with properties that are modified by used operating system features such as: resolution screen, browser history, connected history printers, disk management, applications used, network management, unnecessary files ready for deletion (trash), memory dumps, system activity. All these features are in a freshly installed operating system (in this case, it's a sandbox for recognizing malicious system) set by default as well. These standards are known to the malicious system and can be analyzed as the sandbox [2].

Because of the fact that the use of virtualization is and has been used in several ways, mainly in the production sphere as mentioned above in the chapter about virtual machines, the attackers no longer focus on recognizing the virtual environment but on artifacts environmental wear. If they would focus on recognition of virtual environment many potential victims such as virtualization companies

would miss them. Therefore, this work is further devoted to obfuscation of virtual environment based on creation or simulating wear artifacts.

5. WEAR-AND-TEAR ARTIFACTS

Wear artifacts should be changes in the system resulting from individual adjustments, system impersonation and use of different types of an application. These artifacts can be divided into two groups. Artifacts that arise from direct activity and artifacts that the user does not have direct effect. Artifacts that are directly affected by the user are such as browser artifacts. The URLs that are searched for are captured and downloaded files, or disc artifacts where its files are stored. Based on this information can be found using heuristics user's interests. Artifacts that the user doesn't have for example, some of the registry artifacts in which is an artifact of DLL files that install the software programs. There is some influence between a user and the system because the user downloaded and started the files and which is not direct approach since the user did not directly create those files. Artifacts with direct influence are stronger evidence of wear and tear operating system to users as indirect. Therefore, it is possible that malicious software may assign more weight such artifacts and focus on a wider range of artifacts of this kind. For artifacts that are not directly created users cannot fully rely on optimization, cleaning tools that are in windows systems very popular [11].

Many aspects of the system affect artifacts wear and tear from the operating system. For example, a lot of different file types, network, or responses system for various events and other subsystems. Artifacts wear and tear can be divided into these groups on the basis of subsystems.

- System
- Drive
- Networks
- Registers
- Browsers

The complete table of wear and tear artifacts is located in Table 1

Table 1 Complete list of wear-and-tear artifacts [7]

Category	Name	Description	User	Sandbox	Baseline
System	totalProcesses	# of processes	94.4	35	41
	winupdt	# installed Windows updates	794	19	2
	sysevt	# system of system events	27K	8K	334
	appevt	# of application events	18K	2.4K	184
	syssrc	# sources of system events	78	49	48
	appsrc	# sources of application events	40	26	23
	sysdiffdays	Elapsed time since the first system event (days)	370	1.7K	0
	appdiffdays	Elapsed time since the first application event (days)	298	943	0
Disk	recycleBinSize	Total size of the recycle bin (bytes)	2.5G	50M	0
	recycleBinCount	# files in the recycle bin	109	3	0
	tempFilesSize	Total size of temporary system files (bytes)	302	24	8.2
	tempFilesCount	# temporary system files	411	60	10
	miniDumpSize	Total size of process crash minidump files (bytes)	3M	409K	0
	miniDumpCount	# of process crash minidump files	9	4	0
	thumbsFolderSize	Total size of the system's thumbnails folder (bytes)	63M	8M	2.6M
	desktopFileCount	# files on the desktop	34	6	3
Network	ARPCacheEntries	# entries in the ARP cache	19	4.5	5
	dnscacheEntries	# entries in the DNS resolver cache	151	4	3
	certUtilEntries	# URLs of previously downloaded CRLs	1.7K	210	6
	wirelessnetCount	# of cached wireless SSIDs	8	0	0
	tcpConnections	# of active TCP connections	77	27	16
Registry	regSize	Size of the registry (in bytes)	144.8M	53M	35M
	uninstallCount	# registered software uninstallers	177	58	18
	autoRunCount	# programs that automatically run at system startup	9	3	1
	totalSharedDlls	Legacy DLL reference count	242	176	26
	totalAppPaths	# registered application paths	54	35	23
	totalActiveSetup	# Active Setup application entries	24	32	25
	orphanedCount	# leftover registry entries	11	10	9
	totalMissingDlls	# registered DLLs that do not exist on disk	21	23	13
	usrassistCount	# of entries in the UserAssist cache (frequently opened applications)	222	98	11
	shimCacheCount	# entries in the Application Compatibility Infrastructure (Shim) cache	191K	98K	38K
	MUICacheEntries	# Multi User Interface (MUI) cache entries	66	83	163
	FireruleCount	# of rules in the Windows Firewall	511	332	358
	deviceClsCount	# previously connected USB devices (DeviceInstance IDs)	81	29	60
	USBStorCount	# previously connected USB storage devices	1.7	0	0
Browser	browserNum	# installed browsers (Internet Explorer, Firefox, Chrome)	2.9	1.4	1
	uniqueURLs	# unique visited URLs	28K	13.8	1.64
	totalTypedURLs	# URLs typed in the browser's navigation bar	1.6K	4	1
	totalCookies	# of HTTP cookies	3K	135	2
	uniqueCookieDomains	# unique HTTP cookie domains	1003	23	1
	totalBookmarks	# bookmarks	274	20	1
	totalDownloadedFiles	# downloaded files	340	3	0
	urlDiffDays	Time elapsed between the oldest and newest visited URL (days)	225	370	0
	cookieDiffDays	Time elapsed between the oldest and newest HTTP cookie (days)	303	256	0

5.1. System artifacts

These artifacts are generic systemic features such as the number of running processes; or installed updates are directly linked to system history because more system installed with accumulates over the years. A rich source of information the current state of the system is also recorded events on Windows systems. They are recorded in its different types of system events. Its contents are often application, security, setup and system events of various administrative assessments such as such as critical, erroneous, or warning information; success or system failure reports. Very interesting are the events that are the most common part of these records under normal conditions such as warnings missing files, incorrect network addresses, unexpected shutdowns or rejection errors implementation. Examples of this type of artifacts the wear may be more including various other aspects of different applications that may have records implementation from the past. These artifacts are empty or almost empty only on occasion if the malicious systemin a freshly installed environment, or if the user recently deleted them. And that's why it is not they can be relied upon from the attacker's point of view [11].

5.2. Disc artifacts

The user activity results in a filled file different file types or deleted files or various data in the cache. Most used areas of operating system to ordinary users and are directly user - associated: desktop and application temporary or permanent lubrication called a basket. And finally, it is needed to focus on files that are not created users temporarily or cached data as such as file contents of thumbnails or process such as minidump files [5].

5.3. Network artifacts

Network artifacts are especially from a historical perspective, a strong indicator of current use. Behind everyone when the host is to send a packet to a remote destination, Examines the system's address resolution protocol (ARP) cache to find the physical (MAC) address of the gateway or the address of another host on the local network. Similarly, for each domain name is translated to an IP address, the DNS operating system contains it in the buffer memory of the latest resolutions. Using encryption, the public key is also the result of an artifact associated with the certificates that have been denied. One list certificate denied is downloaded and stored in cache locally. List of URLs previously downloaded rejected lists certificates is taken as another part of the information stored in cache. So, the number of records in this the list is directly dependent on the user and the previous one network activity that automatically creates updates. [9].

5.4. Register artifacts

Windows registers contain a lot of information from different aspects of the system. Some of these artifacts could be mentioned in the previous chapters but because the registers are ranked into one category because it contains information that is not outside of the registers. Whenever a driver or application is installed, windows store key values and pair them in registers. This effect is too extensive. The Windows system is known for following uninstalling applications neglects uninstalling these files and a lot of unused remaining files should be deleted. Therefore, there are many tools that these deletes files save and optimize system files resources. And there are many other artifacts that maybe recognize such as: register size in bytes, count firewall rules because new applications can often install new rules [8].

5.5. Browser artifacts

For many users, browsers have become versatile tool from office supplies respectively. services to internet shopping, banking, games and services social networking or internet surfing. In reality the older the system, the more information the history collects in your browser. It is a direct unusual state that the browser it is empty for a commonly used system. This history the browser is built from several artifacts wear related to user activity. In the morning URLs visited, stored HTTP cookies, saved bookmarks and downloads. Research these artifacts is limited by the user's privacy. Because the user uses multiple browsers for different purposes and most often different as the default browser on the system is set up this one of the artifacts of wear. Because crushing most users use browsers that are not default among the most used operational systems. The first most used browser is Google Chrome with more than 63 percent of users by Globstat. In the second place with more than 15 percent is Safari from Apple and the third most used browser is Firefox [10]. According to the survey, the user has more than 2 browsers in your system. Number of HTTP cookies corresponds to the total number of cookies extracted formal installed browsers [10].

6. DESING AND IMPLEMENTATION OF THE OBFUSCATION TOOL

From system artifacts, we focus on the artifact total processes which is shown in the table above. Straight the user's influence on this artifact is launching applications or processes that these applications create. Next artifacts are network artifacts where the tool focuses on artifacts ARPCacheEntries, dnsCacheEntries and CertUtilEntries. These artifacts are the result of the user work with internet connection and internet browsing. The artifacts of the disk tool can be obfuscated by creating random files, by throwing files into the trash application, linking to applications and files, and saving files to the desktop.

The tool will affect the recycleBinSize, recycleBinCount, tempFileSize artifacts tempFileCount and desktopFilesCount as registers. These artifacts affect the instrument by installing apps. They are regSize artifacts, uninstallCount, totalSharedDlls, totalAppPaths, muiCachedCount. Other artifacts are browser artifacts. These user artifacts influence by crawling URLs and downloading files via the browser. We will influence these actionsbrowserNum artifacts, uniqueURLs, totalTypeURLs, totalCookies, uniqueCookiesDomains, totalDownloadedFiles. For a separate artifact that is not in the table, but it is described in the browser section which

shows the number of crawlers in the system, we download and install the most commonly used browsers by Globstat company statistics. These browsers are Chrome from Google and Firefox from Mozilla [12].

This tool consists of two modules and one main class where the methods of these modules are called. One module provides the file structure and the other the module will be in charge of network traffic. The module file structure will take care of creation of new folders, files, links, install programs and temporarily delete files of different kinds. In the second module we focus on site browsing, file downloads, filling ARP records in registers.

6.1. Creating random files

This part of the module is divided into several steps:

- generation of the file structure paths,
- analysis of the file paths and create an intersection among them,
- files are written with the randomly generated content.

The file structure would be a list character string. One string represents the path to the file. Character strings will consist of randomly selected pre-set options folder names - files. Part of these pre-prepared basic paths are just the most common files used, such as the desktop user. Therefore, this design also affects the artifact desktopFilesCount. Their existence will be verified and based on this, the next interaction is detected. When there is a folder available, we will sink into it and if it does not exist create a new one. Each journey should be unique and count paths will represent the number of files created. The files will work with the base class belonging to basic I/O operations with the name of File.

6.2. Implementation of FileSystemModule module

This module contains several private transformations which are being processed during the obfuscation of file artifacts system. "LinksPaths" variable contains paths or folder names to which links are created in the system. The "windowsPaths" variable contains names the most commonly used folders in the operating system Windows that are an essential part of filesystem of this operating system. Directories variable are pre-defined user file names. "UserFiles" is a variable that contains names of user files. To variable "CreatedFileStructure" there is generated the user file structure from paths that were randomly created by combining previous transformations. These variables have private visibility indicator because they are only used in the module. The variables are of the list type that have generic character string type.

6.3. The createUserFileStucture function

This function does not have a return type. Its task is to create file structure from pre-made lists.

An index of 50 to 100 files is randomly created using class "Random" from library "java.util". We invoke the method "NextInt" from this class that expects a numeric argument. Randomly created number from zero to the entered number creates file paths based on a random number of files from the "windowsPath" and "userFiles" lists. Paths are then added to the "createdUserFileStructure" list.

The deleteRandomFiles function feature uses the user file structure and randomly selects the files to be included in the list on deleted files.

The number of files is randomly generated that move to the Trash app on your operating system. This random index is inserted into the for loop in which randomly selecting files from the list "CreatedUserFileStructure" file. Random file selection is secured by entering a random number into the "get" method that is part of the list object. Follow the path to file creates a File object that represents a link to the real file in the operating system and is inserted into temporary list with generic File type. Chain representing the file path is deleted from the list "CreatedUserFileStructure" to avoid collisions, because the list is still in use and with the files also works in another part of the module. This list contains insert File objects move to Thrash which verifies that an instance of the FileUtils class is available based on "hasThrash" method, which has "boolean" return value if the instance of the FileUtils class has a recycle bin available, so the moveToThrash method is called from fileUtilsobject. The FileUtils class is located in an external libraryJNA. CreateFilesFromPath and generateFileContentThis function gets a string of characters representing file path. This path is divided according to the slash character using the String class method. This method has return type of a string array to store in a temporary variable. Creates a second temporary variable that represents a composite path that a function has already gone through. From the field string selects a string and determines whether it is the last using of the "equals" method from an object string that awaits an argument string type and evaluates whether the string object is inserted into method is equal to the object of which the method was induced. The return type of the method is of the "boolean" type. INif the object is the last of the field that is not attached to its lash.

Otherwise, the slash is assigned an instance of the File class is created. The reference to file or folder exists and if not created in otherwise, it continues. If this is the last object from a string field so a file is created if it is not the last one folder is created. To create a file is used the "write" method from the "Files" class that comes from the libraryJava Nio.

The method has two arguments, one is an object of type "Path" which represents the path to the file and the other is the content we want to fill the created file. This method works with a container that fills according to its second argument into which the tool inserts a value generated by generate File Content. Generation of File Content has a return value of list type chains. It returns a list of strings in a temporary variable which it had declared at the beginning. The function has one argument which is the number of rows. This argument serves as an index for the For cycle. The cycle is generated using the function next Bytes from the object to generate random number of bytes. Feature next Bytes has one argument which is an array of bytes. This the field is filled with random bytes.

6.4. Implementation of the NetworkTrafficModule module

This module contains functions to work with downloaded files, create browser history and creating false arp records. This file downloads module in two ways. Using the Java Nio and using a browser. The fillArpCache and browse functions The "fillArpCache" function fills cache entries using the command line and the "arp -s" command to insert the false arp records argument. The browse function searches pages with instances of the Desktop class from the awt library and invoke methods "Browse". This method starts the search process in default search engine based on its parameter that is URL type.7.

7. CONCLUSIONS

This publication was dedicated to virtual obfuscation systems from a malicious system. The job was to analyze the issue and its connections. The main parts of an obfuscated virtual systems are a virtual system, malicious software and the act of congestion itself. Implementation is needed for long-term purposes artifact obfuscation to expand by more artifacts to make this product more complex. In addition to complexity, the solution needs to be optimized because obfuscation with this tool takes too long and extending this solution could be much more time-consuming.

REFERENCES

[1] STALINGS, W. L. B.: Malicious Software. rev. Computer Security principles and practice, zv. II, Sydney, Austrália: University New South Wales, Australian Defence Force Academy, 2012, pp. 178-219.

[2] MIRAMIRKHANI, N. – , APPINI, M. P. – NIKIFORAKIS, N. – POLYCHRONAKIS, M.:"Spotless Sandboxes: Evading Malware Analysis Systems Using Wear-and-Tear Artifacts," 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, 2017, pp. 1009-1024.

[3] WRIGHT, D. – STROSCHEIN, J.:"A Malware Analysis and Artifact Capture Tool," 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech), Athens, 2018, pp. 328-333.

[4] BRUNTON, F. – NISSENBAUM, H.:"Understanding Obfuscation," in Obfuscation: A User's Guide for Privacy and Protest , , MITP, 2015, pp.44-44.

[5] YOU, I. – YIM, K.:"Malware Obfuscation Techniques: A Brief Survey," 2010 International Conference on Broadband, Wireless Computing, Communication and Applications, Fukuoka, 2010, pp. 297-300.

[6] Red Hat, „www.redhat.com,“ Red Hat, 2019. [Online]. Available: https://www.redhat.com/en/topics/virtualization/what-is-virtualization#.

[7] MIRAMIRKHANI, N. P.: „Spotless Sandboxes: Evading Malware Analysis Systems using Wear-and-Tear Artifacts,“ rev. IEEE Symposium on Security and Privacy, 2017.

[8] YOSHIOKA, K. – HOSOBUCHI, Y. – ORII, T. – MATSUMOTO, T.: "Vulnerability in Public Malware Sandbox Analysis Systems," 2010 10th IEEE/IPSJ International Symposium on Applications and the Internet, Seoul, 2010, pp. 265-268.

[9] INOUE, D. – YOSHIOKA, K. – ETO, M. - HOSHIZAWA, Y. – NALAO, K.: "Automated Malware Analysis System and its Sandbox for Revealing Malware's Internal and External Activities" IEICE Trans. vol. E92D no. 5, 2009.

[10] WILLEMS, C. – HOLZ, T. – FREILING, F.: "Toward Automated Dynamic Malware Analysis Using CWSandbox" Security & Privacy Magazine IEEE vol. 5 no. 2 pp. 32-39, 2007.

[11] BAYER, U. – KRUEGEL, C. – Kirda, E.:"TTAnalyze: A Tool for Analyzing Malware" 15th Annual Conference of the European Institute for Computer Antivirus Research 2006.

[12] PARANTHAMAN, R. – THURAISINGHAM, B.: "Malware Collection and Analysis," 2017 IEEE International Conference on Information Reuse and Integration (IRI), San Diego, CA, 2017, pp. 26-31.

Received October 3, 2019, accepted November 27, 2019

BIOGRAPHIES

Michal Hulič was born on 30.07.1992. In 2017 he graduated (MSc) with distinction at the department of Computers and Informatics of the Faculty of Electrical Engineering and Informatics at Technical University in Košice. Since 2017 he is a PhD student at Department of Computers and Informatics. His scientific research is focusing on computer security.

Martin Chovanec received his Engineering degree in Informatics in 2005 from the Faculty of Electrical Engineering and Informatics, Technical University of Kosice. In 2008 he received his Ph.D. degree at the Department of Computers and Informatics of the Faculty of Electrical Engineering and Informatics of the Technical University of Kosice and his scientific research was focused on network security and encryption algorithms. Currently, he is Director of the Institute of Computer Technology of the Technical University of Kosice.

Viliam Korba received his Bachelor degree in Informatics in 2019 from the Faculty of Electrical Engineering and Informatics, Technical University of Kosice. Currently, He is student the Faculty of Electrical Engineering and Informatics, Technical University of Kosice.