Towards a Robust Method of Dataset Generation of Malicious Activity on a Windows-Based Operating System for Anomaly-Based HIDS Training

Dainius Čeponis and Nikolaj Goranin

Vilnius Gediminas Technical University,
Saulėtekio al. 11, 10223 Vilnius, Lithuania
dainius.ceponis@vgtu.lt, nikolaj.goranin@vgtu.lt

Abstract. Classical cyber-attack detection methods, based on signatures and rules demonstrate stagnation and inability to fight the zero-day, advanced-persistent-threat and similar attacks, while anomaly-based detection methods, although were exploited for a number of years, are still characterized by huge numbers of false-positives (valid user or application behavior, that has been classified as intrusion) and ability to work in relatively stable conditions. The progress achieved in recent years in the area of deep learning artificial intelligence techniques provide a potential for renewing the research on the topic and for achieving promising results. Anomaly-based intrusion detection systems (IDS) utilize the ability to learn from a training set of legal and malicious actions. In order to train anomaly-based IDS systems enormous amount of data is required. Majority of available datasets used for IDS training are related to the network-level based intrusion detection, while datasets for host-based intrusion detection system (HIDS), which is becoming extremely important, training are not available or incomplete and lack important features. In this article we propose a method for automated system-level anomaly dataset generation that is to be used in further training of artificial intelligence-based HIDS training. Details for method implementation are also presented and test results discussed.

Keywords: Anomaly detection, HIDS, Windows system calls

1 Introduction

IDS are systems dedicated for monitoring user and application activity on company networks and computers and alerting on possible attack attempts. More precise definition of intrusion detection system is provided by [1] and defines IDS as “the process of monitoring the events occurring in a Computer system or network and analyzing them for signs of intrusions, defined as attempts to compromise the confidentiality, integrity, availability, or to bypass the security mechanisms of a computer or network”.

Lupeikiene A., Matulevičius R., Vasilecas O. (eds.):
Baltic DB&IS 2018 Joint Proceedings of the Conference Forum and Doctoral Consortium.

IDS can be classified into two types [3]: network-based intrusion detection systems (NIDS) and HDS. HIDS is located on end-point user machine and monitors user and host operating system behavior. HIDS can provide the following functions: file integrity checking, registry monitoring, rootkit detection, policy monitoring, log analyzing and system calls analysis [4]. All that information, in real time, is transferred to the main server. Main server analyses information from the hosts and decides whether to notify security staff on suspicious behavior on the host machine. Suspicious activity can be: abnormal CPU and RAM usage or text editing application attempt to modify system password file. File integrity is also monitored besides activity on the host machine i.e. HIDS can be seen as an agent which monitors system and checks if any other agent violates security policy.

The first intrusion detection model was introduced in 1987 [5]. Three intrusion detection methods have evolved since that time [6]: misuse or signature-based, anomaly-based, and hybrid.

Signature-based detection is designed to detect known attacks and has a small number of false-positives. It scans for patterns associated with known attacks against computer system. Those patterns can be: hardware-related parameters collection (CPU and RAM usage), cryptographic hash value of rootkit or error log generated by an attack [4]. A regular database update on attack pattern is necessary to have a fully functional system. This method is not effective against new (zero-day) attacks until they are added to the database [7].

The idea of anomaly-based intrusion detection is predicated on a belief that an intruder's behavior is noticeably different from that of a legitimate user and that many unauthorized actions are detectable. This type of intrusion detection should be effective against zero-day attacks. Another advantage for this type is that a – detection algorithm can be tailored for a specific company, network or a user, making it challenging task for the attacker to select effective and non-detectable intrusion actions. Numerous machine learning methods of clustering and classification were applied for anomaly detection [8]. The main disadvantage of this type of intrusion detection are the high false positives rates. Large sets of training data are required to construct normal behavior profile [9]. It is also possible that malicious activity is included into the legitimate activity training data – in that case the intrusive activity will be legit in later detection process [10]. Due to this reason it is extremely important to ensure creation of "sterile" datasets that would separate legitimate and malicious actions.

Many IDS systems usually make use of a hybrid method which combines signature and anomaly-based techniques. Such combination provides small amount of false positives for unknown attacks and raises detection rate on known intrusions [11].

So far NIDS systems are dominating the field. However, HIDS systems are receiving more attention due to the fact that they provide more information about intrusion and can prevent from significant damage as well as offering an additional layer of security. However, HIDS research lack suitable datasets that can be used for evaluating new methods. Therefore, we propose a method for training data generation. The paper contains five main sections: related work of datasets generated to train anomaly-based IDS, the proposed method for new dataset generation description, results description, and conclusions.

2 Related Work

Signature-based intrusion detection is showing better results on detecting known attacks, but it fails to report new and unknown attacks. For that reason anomaly-based intrusion detection methods are getting more attention [12, 13] – more than 60% research papers are focused on anomaly detection. However, despite of significant progress in anomaly-based intrusion detection methods, they still show higher false-positive detection rate than signature-based methods. The progress achieved in recent years in the sphere of deep-learning artificial intelligence techniques provide a potential for renewing the research on the topic specified.

A key factor in machine learning, which forms a basis for anomaly detection algorithms, is the quality of data. Most of the recent research on intrusion detection has been done using DARPA and KDD Cup 99 datasets. So far 42 % KDD cup dataset, 20 % DARPA dataset and 38 % other datasets have been used to verify proposed new methods for anomaly detection[13]. KDD Cup 99 dataset was collected in 1999 by processing the tcpdump portions of the 1998 DARPA Intrusion Detection System (IDS) Evaluation dataset, created by Lincoln Lab under contract to DARPA [14, 15]. Those datasets contain various information collected on simulating attacks against a network. Four main attack types have been used against a simulated US Air Force LAN [16]: probing, denial of service attacks, user to root attacks, remote to user attacks.

Therefore, DARPA-related datasets have a data associated to a network and are perfect to apply in NIDS research [17]. It is necessary to stress that DARPA, which has been used as a de facto standard for anomaly-based NIDS training, present the simulated and not the real attack data. Nevertheless, it is still considered by experts as a valuable dataset.

The research done in the sphere of anomaly dataset generation for HIDS training is minimal despite the fact of the growing need for anomaly-based HIDS systems.

Some information has been collected during the KDD dataset assembly. At first it was a UNIX-type host systems information. Later, in January 2000, Windows NT hosts data was collected on similar circumstances [18]. It contains not only tcpdump provided data, but also the Windows NT event log audit data. Despite the provided collections, KDD Cup-related datasets lack host machines-related information and only NIDS researches use it [19].

Some attempts have been made to generate novel public datasets for the Windows operating system. Windows audit logs analysis method was introduced and collected data was prepared for public usage by [20]. Audit logs have been produced by running malware on a target machines. The proposed audit logs analysis method yields high detection rate. Still, audit logs have some disadvantages. One of them – it cannot detect thread injection, which is a main tool in malicious tactics [20].

One of the latest datasets related to the host-based intrusion detection is the ADFA-IDS dataset. In an experiment a zero day attack was simulated and system calls in Windows and Linux operating systems have been collected [19]. Two comprehensive Windows operating system-based datasets (ADFA-WD and ADFA-WD: SAA) were introduced for the research community. Prior to the Windows OS datasets, Linux related collection was introduced [21, 22]. Windows OS datasets contains core dynamic link library (dll) name and called function address. Linux dataset contains sequence of numbers. Those numbers are representing a corresponding system call. All three ADFA-IDS databases (two for Windows and one for Linux) have one big disadvantage - they lack system calls parameters, which could be used to identify specific relations required to perform successful intrusion detection.

The minimal presence of datasets used for HIDS training, and absence of crucial data parameters in these datasets, encourages creation of a simple and reliable malicious activity dataset generation method. Such method must be easily implementable and not dependent on a specific HIDS system.

3 The Proposed Robust Dataset of Malicious Activity Generation Method

3.1 Method Description

The following nonfunctional requirements were formulated for the malicious activity dataset generation method: the system has to be flexible (it must allow adapting new data collection in the future), easy to configure (no special tools must be required to change system parameters), and based on open-source software only. The target operating system for malicious activity collection chosen – was Windows, because it is still the most widely used operating system in the world, although the method can be adapted for any other OS.

For reasons of simplicity and proof of concept, only openly available malware samples were used to generate malicious activity samples. The method can be easily automated: any malware samples can be downloaded, prepared according to the requirements, and used. The proposed method has six following steps:

1. Malware samples preparation. At first, malware must be obtained from available sources. Later, malware of Windows OS executables type should be extracted and added to a separate collection for use.
2. Host machine preparation. Hypervisor must be installed and configured on the selected host machine. Malware samples must be copied to the host machine for later execution.
3. Guest machines preparation. Template for a virtual guest machine must be added and configured on the host machine. Later, the required number of guest machines (copies) should be created for malware execution. Execution of malware samples can be performed in parallel and is dependent on the number of guest machines available.
4. Data collection server preparation. Server storage for the malware execution log information (such as anomaly samples in form of logs, network activity, system calls, etc.) must be prepared.
5. Malware samples execution and data collection. When all machines are prepared – main execution script is started. The main script will upload a malware sample to the target guest machine and will start the operating system. Later, a script on the

guest machine will execute malware on OS startup. Malware-generated activity log will be automatically collected and uploaded to the data collection server.
6. Collected data preparation and analysis. When all samples are executed, collected data can be transformed to the XML format and analyzed.

3.2 Method Implementation

The proposed method implementation (architecture) can be seen in Fig. 1. The virtualization technique, based on a free ProxMox hypervisor, was selected to simulate quest machines that will be used for running malicious actions. ProxMox VE is a completely open-source platform for enterprise virtualization, a built-in web interface that allows management of VMs and containers, software-defined storage and networking, high-availability clustering, and multiple out-of-the-box tools in a single solution [23]. ProxMox is running QEMU - a generic and open source machine emulator and virtualizer and is based on Debian operating system. According to the results of the latest research, QEMU has a less detectable virtualization through basic detection techniques [24], which maximizes the malware execution rate.

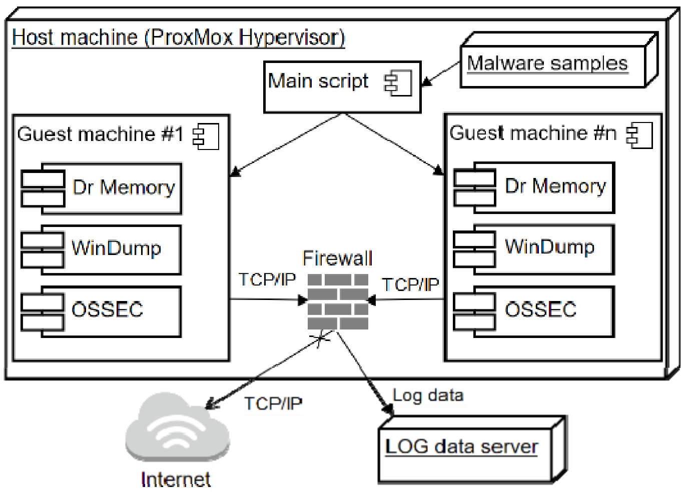

Fig. 1. Malware execution components scheme

A main bash script is executing all commands required to collect data: a guest machine is prepared, started and stopped by that script. The main bash script has only one parameter – a folder that contains prepared malware samples. ProxMox firewall is enabled on the Host machine to manage network flow and minimize the risk of malware propagation. Only one-directional flow to the remote HIDS server was allowed – all other connections were blocked. All data sent to that server was stored on LOG server for later analysis.

An anomaly data collection was done by three tools: Dr. Memory provided system call tracer for the Windows OS, OSSEC (open source HIDS [25]) for file integrity monitoring and WinDump for the network-related information. Dr. Memory tool provides not only system call name, but also passed parameters list and return value. All that information can be used to detect earlier mentioned thread injection, which is missing in method provided by [20].

Open malware collections were used to generate malignant activity on guest machines. Malware execution was conducted on a Windows operating system. For simplicity reasons, during the first step, only malware of executable type was used, in order to minimize dependence on third party applications (e.g. office suites, utilities, viewers or any other). Malware samples were taken from the freely available database provided by VirusShare [26] (For this paper, VirusShare_00289.zip package, created on 2017-05-07, was used) and theZoo [27]. VirusShare provides malware packages in a form of password-protected zip. Every package can contain various types of malicious files that can target different operating systems: Linux, Windows, Mac, Android and iOS. For that reason, each package must be analyzed and only Windows OS-executable malicious samples have been selected in our case. VirusShare samples were combined with theZoo malware collection, that holds most popular and controversial malware samples. theZoo database already contains password protected archives with executable files. Malware sample preparation is presented in Fig 2.

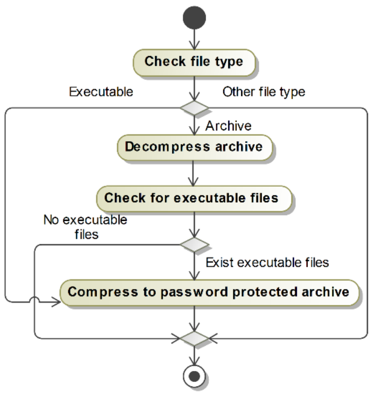

Fig. 2. Malware file preparation

Usually the first byte of a file is holding information about the file type. If that is already a Windows-executable file – corresponding file extension is added to it and the file is packed to the archive with a password “infected”. If the analyzed file is an archive – it is extracted for further analysis and, if executable files are found, they are added to the password protected archive. All other files are skipped.

Malware transfer to the guest machine was implemented with the help of ProxMox VE, which provided the capability to attach an additional virtual drive and copy the file straight to it. It is not dependent on any other third-party software and firewall configuration has no impact on file transfer.

The number of malware samples that can be executed in parallel, thus influencing the dataset generation speed, depends on the number of running guest machines, that is directly related to the available hardware resources. For our experiments tests were performed on the HP ProLiant DL 380 G6 server with the following specifications: 2x Xeon E5520 CPU, 8 GB of RAM and 4x146 GB HDD's connected to RAID 5. Six guest machines were running in parallel.

A bash script on the host machine was used to control guest machine's state (startup and shutdown) and malicious file transfer to the corresponding virtual drive. Virtual drive preparation for the guest machine also was implemented via bash script: it can be mounted on a hypervisor system and updated with required malware file. Main actions performed by the bash script on the host machine are:

1. Copying guest machines disks from prepared templates.
2. Mounting virtual disk for every guest machine, copying prepared malware, unmounting disk.
3. Starting the guest machine.
4. Pausing script for defined time to provide the malware the possibility to reveal all functionality and features. The default pause time in tests was equal to 30 minutes.
5. Stopping the guest machines. The Stop command will halt the machine immediately. Shutdown process is not initiated.

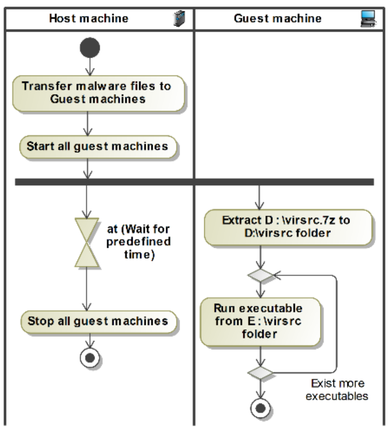
Host machine
Guest machine
Transfer malware files to Guest machines
Start all guest machines
at (Wait for predefined time)
Stop all guest machines
Extract D : \virsrc.7z to D:\virsrc folder
Run executable from E : \virsrc folder
Exist more executables

Fig. 3. Activity diagram of single malware sample batch execution

Guest machine images were also prepared. Each guest machine was running Windows 7 OS and Dr. Memory, OSSEC agent and WinDump. A malware execution script was added to the Windows task scheduler. Task scheduler provides all required privileges for an unimpeded application/malware startup. Then defined archive file is extracted, malware is executed by a run command for every executable in the extracted folder. The anomaly data gathered (list of modified/accessed files, system calls with related information and network data) was sent to the LOG server for analysis.

All actions required for implementing malware samples execution are presented in Fig. 3. Malware samples are executed in a batch manner. Every batch has a number of files identical to the number of available guest machines. It can be seen, that host machine waits for the predefined time while a script on guest machine is executing the provided malware sample. This pause is needed to collect anomaly activities in case malware manifests itself after some delay after infecting the machine.

4 Results And Discussion

A total of 12226 executable malware samples in the form of password protected archives were prepared from theZoo and VirusShare provided packages and used for tests. All samples were tested over the period of two months (2017.07-2017.08). No interruptions or errors related to the malware execution were noticed which is an advantage against well-known tool for such task - Cuckoo sandbox. According to Miller et al. – it has stability issues that cause Cuckoo samples results to be inconsistent between runs [24].

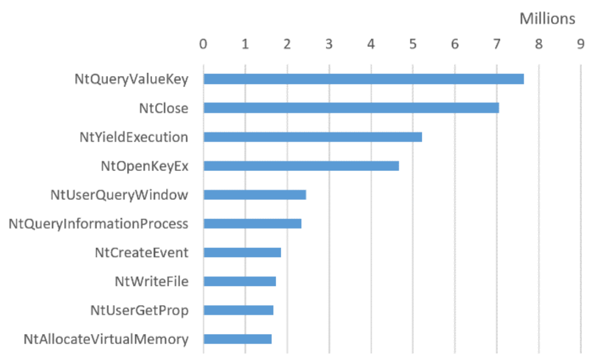

Fig. 4. Most frequently requested system calls

112.56 million system calls traces generated by malware samples were recorded to the database. That amount of data had a massive impact on the database size – generated SQLite file consumes 39.1 GB of storage space. The database contains not only extended system calls information (parameters list, return values) but also metadata about the malware. All that information was imported with the help of Academic API provided by Virus Total. Information for every malware record has included: information of detected malware, positive scan results value, web page to malware description page, malware behavior information (file system action, network communication, loaded modules (dll files)). In our tests that were performed on the basis of Windows 7 OS 645 distinct system calls were captured. The most commonly used system calls are presented on Fig. 4. The dominating part of calls generated by malware were related to registry querying. The next dominating group of calls was implementing the file processing functions (reading and writing).

The collected data has new valuable attributes that can be used to train anomaly-based HIDS system properly. It is necessary to stress that as with any ML methods, the anomaly-based IDS trained under dataset generated, will be able to detect malware possessing features that were presented in the dataset.

5 Conclusions

The performed analysis has shown that there is an increasing requirement for the development and training of anomaly-based HIDS solutions, which is currently being slowed down due to the lack of available and suitable host-level anomaly datasets.

The method for host-level anomaly dataset generation was proposed. The proposed method is based on malware execution in a sterile, isolated virtual machine environment with further anomaly activity collection and data representation in SQLite database format.

The method was implemented and tested only with free or open-source tools and freely available malware samples. The tests performed have proved the method stability and method suitability for host-level anomaly dataset generation. Automated anomaly generation allows flexible training data-set expansion, response to the new attack types and generation of specific on-demand datasets.

Anomaly database of system call traces and other anomaly data was generated for 12226 malware samples. The dataset generated using the method proposed has an advantage against existing datasets because of additional parameters (system call arguments list and return value) that allow more in-depth HIDS training.

An expansion of the generated dataset is being planned for creating a more comprehensive host-level anomalies dataset for HIDS training. The expansion is planned via inclusion of non-executive type malware samples, non-malware attacks and optimizing the pause interval for better feature assembly of delayed malware activities.

References

1. Bace, R., Mell, P.: NIST Special Publication on Intrusion Detection Systems (2001)
2. Bhattacharyya, D.K., Kalita, J.K.: Network Anomaly Detection: A Machine Learning Perspective. Chapman and Hall/CRC (2013)
3. Garcia-Teodoro, P., Diaz-Verdejo, J., Macia-Fernandez, G., Vazquez, E.: Network anomaly detection: methods, systems and tools. Comput. Secur. 28, 18-28 (2009). doi:http://dx.doi.org/10.1016/j.cose.2008.08.003
4. Hay, A., Cid, D., Bary, R., Northcutt, S.: OSSEC Host-Based Intrusion Detection Guide (2008)
5. Denning, D.E.: An intrusion-detection model. IEEE Trans. Softw. Eng. 13, 222-232 (1987). doi:10.1109/TSE.1987.232894
6. Buczak, A.L., Guven, E.: A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Commun. Surv. TUTORIALS 18 (2016). doi:10.1109/COMST.2015.2494502
7. Xie, M., Hu, J.: Evaluating host-based anomaly detection systems: A preliminary analysis of ADFA-LD. In: Proc. 2013 6th Int. Congr. Image Signal Process. CISP 2013, 3, pp. 1711-1716 (2013). doi:10.1109/CISP.2013.6743952
8. Agrawal, S., Agrawal, J.: Survey on anomaly detection using data mining techniques. In: Procedia Computer Science (2015)
9. Aydin, M.A., Zaim, A.H., Ceylan, K.G.: A hybrid intrusion detection system design for computer network security. Comput. Electr. Eng. 35, 517-526 (2009). doi:10.1016/j.compeleceng.2008.12.005

10. Tan, K.M.C., Killourhy, K.S., Maxion, R.A.: Undermining an anomaly-based intrusion detection system using common exploits. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pp. 54-73 (2002)
11. Depren, O., Topallar, M., Anarim, E., Ciliz, M.K.: An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks. Expert Syst. Appl. 29, 713-722 (2005). doi:10.1016/j.eswa.2005.05.002
12. Hu, J.: Host-based anomaly intrusion detection. Handb. Inf. Commun. Secur., 235-255 (2010)
13. Azad, C., Jha, V.K.: Data mining in intrusion detection: a comparative study of methods, types and data sets. Int. J. Inf. Technol. Comput. Sci. 5, 75-90 (2013). doi:10.5815/ijitcs.2013.08.08
14. Lippmann, R.P., Fried, D.J., Graf, I., Haines, J.W., Kendall, K.R., McClung, D., Weber, D., Webster, S.E., Wyschogrod, D., Cunningham, R.K., Zissman, M. A.: Evaluating intrusion detection systems without attacking your friends: The 1998 DARPA intrusion detection evaluation. In: DARPA Inf. Surviv. Conf. Expo. 2000. DISCEX '00. Proc., pp. 12-26, vol.2 (1999). doi:10.1109/DISCEX.2000.821506
15. Brugger, T.: KDD Cup'99 dataset (Network Intrusion) considered harmful. KDnuggets Newsl. 7, 15 (2007)
16. Mukkamala, S., Sung, A.H., Abraham, A.: Intrusion detection using an ensemble of intelligent paradigms. J. Netw. Comput. Appl. 28, 167-182 (2005). doi:10.1016/j.jnca.2004.01.003
17. Sahu, S.K., Sarangi, S., Jena, S.K.: A detail analysis on intrusion detection datasets. In: Souvenir 2014 IEEE Int. Adv. Comput. Conf. IACC 2014, pp. 1348-1353 (2014). doi:10.1109/IAdCC.2014.6779523
18. Korba, J.: Windows NT Attacks for the Evaluation of Intrusion Detection Systems* Windows NT Attacks for the Evaluation of Intrusion Detection Systems (2000)
19. Haider, W., Creech, G., Xie, Y., Hu, J.: Windows based data sets for evaluation of robustness of host based Intrusion Detection Systems (IDS) to zero-day and stealth attacks. Futur. Internet., 8, (2016). doi:10.3390/fi8030029
20. Berlin, K., Slater, D., Saxe, J.: Malicious behavior detection using windows audit logs. In: Proc. 8th ACM Work. Artif. Intell. Secur. - AISec '15, 35-44 (2015). doi:10.1145/2808769.2808773
21. Creech, G.: Developing a high-accuracy cross platform Host-Based Intrusion Detection System capable of reliably detecting zero-day attacks. 215 (2014)
22. Creech, G., Hu, J.: Generation of a new IDS test dataset: Time to retire the KDD collection. In: IEEE Wirel. Commun. Netw. Conf. WCNC, pp. 4487-4492 (2013). doi:10.1109/WCNC.2013.6555301
23. Kovari, A., Dukan, P.: KVM & OpenVZ virtualization based IaaS open source cloud virtualization platforms: OpenNode, Proxmox VE. In: IEEE 10th Jubil. Int. Symp. Intell. Syst. Informatics, SISY 2012, pp. 335–339 (2012). doi:10.1109/SISY.2012.6339540
24. Miller, C., Glendowne, D., Cook, H., Thomas, D., Lanclos, C., Pape, P.: Insights gained from constructing a large scale dynamic analysis platform. Digit. Investig. 22, 48-56 (2017). doi:10.1016/j.diin.2017.06.007
25. Timofte, J.: Intrusion Detection using Open Source Tools. Architecture. 2, 75–79 (2008)
26. VirusShare.com: VirusShare.com, https://virusshare.com/, last accessed 2018/04/05.
27. thezoo.morirt.com: theZoo aka Malware DB by ytisf, http://thezoo.morirt.com/, last accessed 2018/04/05.