Journal of Information Security, 2014, 5, 56-64
Published Online April 2014 in SciRes. http://www.scirp.org/journal/jis
http://dx.doi.org/10.4236/jis.2014.52006

Malware Analysis and Classification: A Survey

Ekta Gandotra, Divya Bansal, Sanjeev Sofat

Department of Computer Science and Engineering, PEC University of Technology, Chandigarh, India
Email: ekta.gandotra@gmail.com, divya@pec.ac.in, sanjeevsofat@pec.ac.in

Received 21 February 2014; revised 21 March 2014; accepted 28 March 2014

Abstract

One of the major and serious threats on the Internet today is malicious software, often referred to as a malware. The malwares being designed by attackers are polymorphic and metamorphic which have the ability to change their code as they propagate. Moreover, the diversity and volume of their variants severely undermine the effectiveness of traditional defenses which typically use signature based techniques and are unable to detect the previously unknown malicious executables. The variants of malware families share typical behavioral patterns reflecting their origin and purpose. The behavioral patterns obtained either statically or dynamically can be exploited to detect and classify unknown malwares into their known families using machine learning techniques. This survey paper provides an overview of techniques for analyzing and classifying the malwares.

Keywords

Malware; Static Analysis; Dynamic Analysis; Machine Learning; Classification; Clustering

1. Introduction

Software that "deliberately fulfills the harmful intent of an attacker" is referred to as malicious software or malware [1]. These are intended to gain access to computer systems and network resources, disturb computer operations, and gather personal information without taking the consent of system's owner, thus creating a menace to the availability of the internet, integrity of its hosts, and the privacy of its users. Malwares come in wide range of variations like Virus, Worm, Trojan-horse, Rootkit, Backdoor, Botnet, Spyware, Adware etc. These classes of malwares are not mutually exclusive meaning thereby that a particular malware may reveal the characteristics of multiple classes at the same time.

Malware is one of the most terrible and major security threats facing the Internet today. According to a survey,

How to cite this paper: Gandotra, E., *et al.* (2014) Malware Analysis and Classification: A Survey. *Journal of Information Security*, **5**, 56-64. http://dx.doi.org/10.4236/jis.2014.52006

[2] conducted by FireEye in June 2013, 47% of the organizations experienced malware security incidents/network breaches in the past one year. The malwares are continuously growing in volume (growing threat landscape), variety (innovative malicious methods) and velocity (fluidity of threats) [3]. These are evolving, becoming more sophisticated and using new ways to target computers and mobile devices. McAfee [4] catalogs over 100,000 new malware samples every day means about 69 new threats every minute or about one threat per second. With the increase in readily available and sophisticated tools, the new generation cyber threats/attacks are becoming more targeted, persistent and unknown. **Figure 1** depicts the comparison of traditional and advanced malwares. The advanced malwares are targeted, unknown, stealthy, personalized and zero day as compared to the traditional malwares which were broad, known, open and one time. Once inside, they hide, replicate and disable host protections. After getting installed, they call their command and control servers for further instructions, which could be to steal data, infect other machines, and allow reconnaissance [5].

Attackers exploit vulnerabilities in web services, browsers and operating systems, or use social engineering techniques to make users run the malicious code in order to spread malwares. Malware authors use obfuscation techniques [6] like dead code insertion, register reassignment, subroutine reordering, instruction substitution, code transposition, and code integration to evade detection by traditional defenses like firewalls, antivirus and gateways which typically use signature based techniques and are unable to detect the previously unseen malicious executables. Commercial antivirus vendors are not able to offer immediate protection for zero day malwares as they need to analyze these to create their signatures.

To overcome the limitation of signature based methods, malware analysis techniques are being followed, which can be either static or dynamic. The malware analysis techniques help the analysts to understand the risks and intensions associated with a malicious code sample. The insight so obtained can be used to react to new trends in malware development or take preventive measures to cope with the threats coming in future. Features derived from analysis of malware can be used to group unknown malwares and classify them into their existing families. This paper presents a review of techniques/approaches for analyzing and classifying the malware executables.

2. Malware Analysis

Before creating the signatures for newly arrived malwares, these are required to be analyzed so as to understand the associated risks and intensions. The malicious program and its capabilities can be observed either by examining its code or by executing it in a safe environment.

2.1. Static Analysis

Analyzing malicious software without executing it is called static analysis. The detection patterns used in static analysis include string signature, byte-sequence n-grams, syntactic library call, control flow graph and opcode (operational code) frequency distribution etc. The executable has to be unpacked and decrypted before doing static analysis. The disassembler/debugger and memory dumper tools can be used to reverse com-

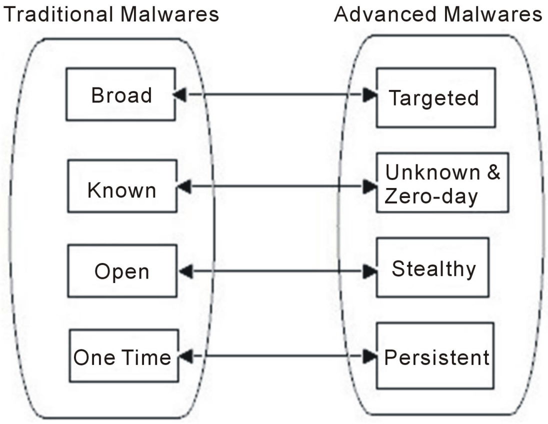

Figure 1. Traditional vs. advanced malwares.

pile windows executables. Disassemble/Debugger tools like IDA Pro [7] and OllyDbg [8] displays the malware's code as Intel ×86 assembly instructions, which provide a lot of insight into what the malware is doing and provide patterns to identify the attackers. Memory dumper tools like LordPE [9] and OllyDump [10] are used to obtain protected code located in the system's memory and dump it to a file. This is a useful technique to analyze packed executables which are difficult to disassemble.

Binary obfuscation techniques, which transform the malware binaries into self compressed and uniquely structured binary files, are designed to resist reverse engineering and thus make the static analysis very expensive and unreliable. Moreover, when utilizing binary executables (obtained by compiling source code) for static analysis, the information like size of data structures or variables gets lost thereby complicating the malware code analysis [11]. The evolving evasion techniques being used by malware writers to thwart static analysis led to the development of dynamic analysis. Moser *et al*. [12], explored the drawbacks of static analysis methodology. In their work, they introduced a scheme based on code obfuscation revealing the fact that the static analysis alone is not enough to detect or classify malwares. Further, they proposed that dynamic analysis is a necessary complement to static analysis as it is less vulnerable to code obfuscation conversion.

2.2. Dynamic Analysis

Analyzing the behavior of a malicious code (interaction with the system) while it is being executed in a controlled environment (virtual machine, simulator, emulator, sandbox etc.) is called dynamic analysis. Before executing the malware sample, the appropriate monitoring tools like Process Monitor [13] and Capture BAT [14] (for file system and registry monitoring), Process Explorer [15] and Process Hackerreplace [16] (for process monitoring), Wireshark [17] (for network monitoring) and Regshot [18] (for system change detection) are installed and activated.

Various techniques that can be applied to perform dynamic analysis include function call monitoring, function parameter analysis, information flow tracking, instruction traces and autostart extensibility points etc. [11]. Dynamic analysis is more effective as compared to static analysis and does not require the executable to be disassembled. It discloses the malwares' natural behavior which is more resilient to static analysis. However, it is time intensive and resource consuming, thus elevating the scalability issues. The virtual environment in which malwares are executed is different from the real one and the malwares may perform in different ways resulting in artificial behavior rather than the exact one. In addition to this, sometimes the malware behavior is triggered only under certain conditions (on specific system date or via a specific command) and can't be detected in virtual environment. Several online automated tools exist for dynamic analysis of malwares, e.g. Norman Sandbox [19], CWSandbox [20], Anubis [21] and TTAnalyzer [22], Ether [23] and ThreatExpert [24]. The analysis reports generated by these tools give in-depth understanding of the malware behavior and valuable insight into the actions performed by them. The analysis system is required to have an appropriate representation for malwares, which are then used for classification either based on similarity measure or feature vectors.

However a large number of new malware samples arriving at anti-virus vendors every day requires an automated approach so as to limit the number of samples that require close human analysis. Several Artificial Intelligence techniques, particularly machine-learning based techniques have been used in the literature for automated malware analysis and classification.

3. Machine Learning for Detecting and Classifying Malwares

Various machine learning approaches like Association Rule, Support Vector Machine, Decision Tree, Random Forest, Naive Bayes and Clustering have been proposed for detecting and classifying unknown samples into either known malware families or underline those samples that exhibit unseen behavior, for detailed analysis. A few of these used in the literature are discussed in this section.

Schultz *et al*. [25] were the first to introduce the concept of data mining for detecting malwares. They used three different static features for malware classification: Portable Executable (PE), strings and byte sequences. In the PE approach, the features (like list of DLLs used by the binary, the list of DLL function calls, and number of different system calls used within each DLL) are extracted from DLL information inside PE files. Strings are extracted from the executables based on the text strings that are encoded in program files. The byte sequence approach uses sequences of n bytes extracted from an executable file. They used a data set consisted of 4266 files including 3265 malicious and 1001 benign programs. A rule induction algorithm called Ripper [26] was applied to find patterns in the DLL data. A learning algorithm Naive Bayes was used to find patterns in the string data and n-grams of byte sequences were used as input data for the Multinomial Naive Bayes algorithm. The Naive Bayes algorithm, taking strings as input data, gives the highest classification accuracy of 97.11%. The authors claimed that the rate of detection of malwares using data mining method is twice as compared to signature based method.

Later on their results were improved by Kolter *et al*. [27]. They used n-gram (instead of non-overlapping byte sequence) and data mining method to detect malicious executables. They used different classifiers including Naive-Bayes, Support Vector Machine, Decision Tree and their boosted versions. They concluded that boosted decision tree gives the best classification results.

Nataraj *et al*. [28] proposed a method for visualizing and classifying malwares using image processing techniques, which visualize malware binaries as gray-scale images. A K-nearest neighbor technique with Euclidean distance method is used for malware classification. Though it is very fast method as compared to other malware analysis methods, the limitation is that an attacker can adopt countermeasures to beat the system because this method uses global image based features. For example, an attacker could relocate sections in a binary or add vast amount of redundant data. In [29], the authors compared binary texture based analysis (based on image processing technique) with that of dynamic analysis. They found that classification using this method is faster, scalable and is comparable to dynamic analysis in terms of accuracy. They also found that this approach can robustly classify large number of malwares with both packed and unpacked samples. The limitation is that this method is vulnerable to knowledgeable adversaries who can obfuscate their malicious code to defeat texture analysis.

Kong *et al*. [30] presented a framework for automated malware classification based on structural information (function call graph) of malwares. After extracting the fine grained features based on function call graph for each malware sample, the similarity is evaluated for two malware programs by applying discriminate distance metric learning which clusters the malware samples belonging to same family while keeping the different clusters separate by a marginal distance. The authors then used an ensemble of classifiers that learn from pair wise malware distances to classify malwares into their respective families.

Tian *et al*. [31] used function length frequency to classify Trojans. Function length is measured by the number of bytes in the code. Their results indicate that the function length along with its frequency is significant in identifying malware family and can be combined with other features for fast and scalable malware classification. Further they noted that usually an obfuscated file does not have any string consisting of words or sentences and thus used printable string information contained within the executables [32]. They used machine learning algorithms available in WEKA [33] library for classifying malwares.

Santos *et al*. [34] pointed out that supervised learning requires a significant amount of labeled executables for both classes (malicious as well as benign datasets) and proposed a semi-supervised learning approach for detecting unknown malwares. It is designed to build a machine learning classifier using a lot of labeled and unlabelled instances. A semi-supervised algorithm LLGC (Learning with Local and Global Consistency) is used, which is able to learn from labeled and unlabelled data and provides a solution with respect to the intrinsic structure displayed by both labeled and unlabelled instances. Executables are represented by using n-gram distribution technique. They also determine and evaluate the optimal number of labeled instances and effect of this parameter on the accuracy of the model. The main contribution of this research is to reduce the number of required labeled instances while maintaining high precision. The limitation is that the previous supervised learning approaches presented in [27] and [35] obtain better results (above 90% of accuracy) than the presented semi-supervised approach. Further in [36], the authors proposed a collective learning approach to detect unknown malwares. It is a type of semi-supervised learning that presents the method for optimizing the classification of partially-labeled data. Collective classification algorithms are used to build different machine learning classifiers using a set of labeled and unlabelled instances. It is validated that the labeling efforts are lower than when supervised learning is used while maintaining the high accuracy rate.

Siddiqui *et al*. [37] used variable length instruction sequence along with machine learning for detecting worms in the wild. Before disassembling the files, they detect compilers, packers. Sequence reduction was done and decision tree and random forest machine learning models were used for classification. They tested their method on a data set of 2774 including 1444 worms and 1330 benign files.

Many researchers now prefer to work on dynamic techniques so as to improve the accuracy and effectiveness of malware classification.

Zolkipli *et al*. [38] presented an approach for malware behavior analysis. They used HoneyClients and Amun as security tools to collect malwares. Behaviors of these malwares are identified by executing every sample on both CWSandbox [20] and Anubis [21] on virtual machine platform. The results generated by both of these analyzers are customized using human based behavior analysis. Then the malwares are grouped into malware families Worms and Trojans. The limitation of this work is that customization using human analysis is not possible for today's real time traffic which is voluminous and having a variety of threats.

Rieck *et al*. [39] proposed a framework for automatic analysis of malware behavior using machine learning. This framework collected large number of malware samples and monitored their behavior using a sandbox environment. By embedding the observed behavior in a vector space, they apply the learning algorithms. Clustering is used to identify the novel classes of malware with similar behavior. Assigning unknown malware to these discovered classes is done by classification. Based on both, clustering and classification, an incremental approach is used for behavior-based analysis, capable of processing the behavior of thousands of malware binaries on daily basis.

Anderson *et al*. [40] presented a malware detection algorithm based on the analysis of graphs constructed from dynamically collected instruction traces. A modified version of Ether malware analysis framework [23] is used to collect data. The method uses 2-grams to condition the transition probabilities of a markov chain (treated as a graph). Machinery of graph kernels is used to construct a similarity matrix between instances in the training set. Kernel matrix is constructed by using two distinct measures of similarity: a Gaussian kernel, which measures local similarity between the graph edges and a spectral kernel which measures global similarity between the graphs. From the kernel matrix, a support vector machine is trained to classify the test data. The performance of multiple kernel learning method used in this work is demonstrated by discriminating different instances of malware and benign software. Limitation of this approach is that the computation complexity is very high, thus limiting its use in real world setting.

Bayer *et al*. [41] proposed a system that clusters large sets of malicious binaries based on their behavior effectively and automatically. The proposed technique relies on Anubis [21] to generate execution traces of all the samples. Anubis was extended in this work with taint-propagation capabilities, to make use of additional information sources. After creating the extraction traces along with taint information, a behavioral profile is extracted for each trace, which serves as input to the clustering algorithm. The clustering algorithm used is based on Locality Sensitive Hashing (LSH), [42] which is a sub linear (efficient) approach to the approximate nearest neighbor problem. LSH can be used to perform an approximate clustering while computing only a small fraction of the $n^2/2$ distances between pairs of points. The authors demonstrate the scalability of their approach by clustering a set of 75,000 malware samples in three hours.

In [43], Tian *et al*. used an automated tool for extracting API call sequences from executables while these are running in a virtual environment. They used the classifiers available in WEKA library [33] to discriminate malware files from clean files as well as for classifying malwares into their families. They used a data set of 1368 malwares and 456 cleawares to demonstrate their work and achieved an accuracy of over 97%.

Biley *et al*. [44] pointed out that the antivirus (AV) products characterize the malwares in ways that are not consistent across various AV products, not complete across malwares and are not concise in their semantics. They developed a classification technique that describes malwares' behavior in terms of system state changes. Binaries are executed in virtualized environment with windows XP installed. The virtual machine is partially firewalled to limit the impact of any immediate attack behavior during the execution period. A behavioral fingerprint of malware's activity is created which includes files written, processes created and network connection etc. A pair wise single linkage hierarchical clustering of the fingerprints using normalized compression distance (NCD) as a distance metric is used to cluster the malwares. The technique is applied to the automated classification and analysis of 3700 malware samples collected over a period of six months. They also measure and compare the consistency, completeness and conciseness of the clusters with that of AV products. The limitation of this work is that the capability and environment of the virtualized system is static throughout the experiments for consistency.

Park *et al*. [45] proposed a malware classification method which is based on maximal component subgraph detection. After executing the malware samples in sandboxed environment, system calls along with parameter values of these calls are captured and a directed graph is generated from these system call traces. The maximal common subgraph is computed to compare two programs. The drawback of this method is that there are some known malware samples that manage to gain kernel-mode privileges without making use of system call interface and can evade the analysis method.

Firdausi *et al*. [46] presented a proof of concept of a malware detection method. Firstly the behavior of malware samples is analyzed in sandbox environment using Anubis [21]. The reports generated are preprocessed into sparse vector models for classification using machine learning. The performance comparison of 5 different classifiers *i.e*. *k*-Nearest Neighbors (*k*NN), Naive Bayes, J48 Decision Tree, Support Vector Machine (SVM), and Multilayer Perceptron Neural Network (MLP) is done on a small data set of 220 malicious samples and 250 benign samples with and without feature selection. The obtained results depicted that overall best performance is achieved by J48 decision tree with a recall of 95.9%, a false positive rate of 2.4%, a precision of 97.3%, and an accuracy of 96.8%.

Nari *et al*. [47] presented a framework for automated malware classification into their respective families based on network behavior. Network traces are taken as input to the framework in the form of pcap files, from which the network flows are extracted. Then a behavior graph is created to represent the network activities of malwares and dependencies between network flows. From these behavior graphs, the features like graph size, root out-degree, average out-degree, maximum out-degree, number of specific nodes are extracted. These features are then used to classify malwares using classification algorithms available in WEKA library [33] and it is concluded that J48 decision tree performs better than other classifiers.

Lee *et al*. [48] proposed a method that clusters the malicious programs by using machine learning method. All the samples of data set are executed in a virtual environment and system calls along with their arguments are monitored. A behavioral profile is created on the basis of information recorded regarding sample's interaction with system resources like registry keys, writing files and network activities. The similarity between two profiles is calculated and then by applying k-medoids, different samples are grouped into different clusters. After completing the training process, the new and unknown samples are assigned to the cluster having medoid closer to the sample *i.e*. nearest neighbor.

It is clear that a single view either static or dynamic is not sufficient for efficiently and accurately classifying malicious programs because of the obfuscation and execution-stalling techniques. So, researches have adapted a hybrid technique which incorporates both static and dynamic features simultaneously for better malware detection and classification.

Santos *et al*. [49] proposed a hybrid unknown malware detector called OPEM, which utilizes a set of features obtained from both static and dynamic analysis of malicious code. The static features are obtained by modeling an executable as a sequence of operational codes and dynamic features are obtained by monitoring system calls, operations and raised exceptions. The approach is then validated over two different data sets by considering different learning algorithms for classifiers Decision Tree, K-nearest neighbor, Bayesian network, and Support Vector Machine and it has been found that this hybrid approach enhances the performance of both approaches when run separately.

A similar work is done by Islam *et al*. [50] to classify the executables into malicious and benign files using both static and dynamic features. The static features used in this work include function length frequency and printable sting information and dynamic features used are API function names and API parameters. The experiment was conducted using 2939 executable files including 541 clean files separately for every feature *i.e*. function length frequency, printable string information and API function calls and then for integrated method for meta classifiers SVM, IB1, DT and RF. The obtained results showed that all meta-classifiers achieve highest accuracy for integrated features and meta-RF is the best performer for all cases. The authors also compared their integrated method accuracy with those of the existing ones and found that their approach is showing the best results.

Anderson *et al*. [51] proposed a method, in which multiple data sources (the static binary, the disassembled binary file, its control flow graph, a dynamic instruction trace & system call trace, and a file information feature vector) are used. For the binary file, disassembled file, and two dynamic traces, kernels based on the Markov chain graphs are used. For the control flow graph, a graphlet kernel is used and for the file information feature vector, a standard Gaussian kernel is used. Then multiple kernel learning is employed to find a weighted combination of the data sources and support vector machine classifier is used to classify the dataset into malicious and benign. It is tested on a dataset of 780 malware and 776 benign instances giving an accuracy of 98.07%.

Data mining and machine learning techniques are being used since long for detecting and classifying malwares. Today internet traffic generates a huge data sets (ranging to peta-scales) for daily monitoring and pattern

Table 1. A summary of publications pertaining to malware analysis and classification.

Malware analysis technique	References
Static	[25] [27] [28] [30] [31] [32] [34] [35] [37]
Dynamic	[38] [39] [40] [41] [43] [44] [45] [46] [47] [48]
Hybrid	[49] [50] [51]

profiling in cyber security models and requires scalable, fast and flexible machine learning techniques. A huge amount of network data is streaming continuously and dynamically in cyber infrastructure which challenges machine learning modeling and requires new principles, methodologies and algorithms for transforming raw data into the useful information.

4. Conclusion

Advanced malwares are posing a severe threat to the internet and computer systems. Signature-based antivirus products are able to detect only those malwares that has already caused damage and are registered. Ever evolving techniques used by malicious softwares to thwart static analysis directed to the development of dynamic analysis which executes the malware sample under controlled environment and monitors its behavior. The reports generated by dynamic analysis can be compiled into behavioral profiles that can be clustered to combine samples with similar behavior into coherent families. The disadvantage of dynamic analysis is that it is time and resource intensive. To counter the trade-off between analysis speed and detecting obfuscated malwares, researches have adapted a technique incorporating a combination of static and dynamic features for detecting and classifying malwares. This paper highlights the existing techniques for analyzing, detecting and classifying malwares. **Table 1** gives the list of publications categorized according to malware analysis techniques. The machine-learning technologies that are being used in detecting and classifying malwares are not adequate to handle challenges arising from the huge amount of dynamic and severely imbalanced network data. These should be transformed so that their potential can be leveraged to address the challenges posed in cyber security.

Acknowledgements

This work is done in Cyber Security Research Center (CSRC) located at PEC University of Technology, Chandigarh, India. The authors would like to thank Government of India, Ministry of Communications and Information Technology, Department of Information Technology, New Delhi, for funding the Project "**Development of Cloud Based Framework for Delivering Security as a Service**", under which this research work has been carried out.

References

[1] Bayer, U., Moser, A., Kruegel, C. and Kirda, E. (2006) Dynamic Analysis of Malicious Code. *Journal in Computer Virology*, **2**, 67-77. http://dx.doi.org/10.1007/s11416-006-0012-2

[2] (2013) The Need for Speed: 2013 Incident Response Survey, FireEye. http://www.inforisktoday.in/surveys/2013-incident-response-survey-s-18

[3] (2012) Addressing Big Data Security Challenges: The Right Tools for Smart Protection. http://www.trendmicro.com/cloud-content/us/pdfs/business/white-papers/wp_addressing-big-data-security-challenges.pdf

[4] (2013) Infographic: The State of Malware. http://www.mcafee.com/in/security-awareness/articles/state-of-malware-2013.aspx

[5] (2013) Next Generation Threats. http://www.fireeye.com/threat-protection/

[6] You, I. and Yim, K. (2010) Malware Obfuscation Techniques: A Brief Survey. *Proceedings of International conference on Broadband, Wireless Computing, Communication and Applications*, Fukuoka, 4-6 November 2010, 297-300. http://dx.doi.org/10.1109/BWCCA.2010.85

[7] IDAPro. https://www.hex-rays.com/products/ida/support/download_freeware.shtml

[8] OllyDbg. http://www.ollydbg.de/

[9] LordPE. http://www.woodmann.com/collaborative/tools/index.php/LordPE

[10] OllyDump. http://www.woodmann.com/collaborative/tools/index.php/OllyDump

[11] Egele, M., Scholte, T., Kirda, E. and Kruegel, C. (2012) A Survey on Automated Dynamic Malware-Analysis Techniques and Tools. *Journal in ACM Computing Surveys*, **44**, Article No. 6.

[12] Moser, A., Kruegel, C.and Kirda, E. (2007) Limits of Static Analysis for Malware Detection. 23*rd Annual Computer Security Applications Conference*, Miami Beach, 421-430.

[13] (2014) Process Monitor. http://technet.microsoft.com/en-us/sysinternals/bb896645.aspx

[14] Capture BAT. https://www.honeynet.org/node/315

[15] (2014) Process Explorer. http://technet.microsoft.com/en-us/sysinternals/bb896653.aspx

[16] Process Hackerreplace. http://processhacker.sourceforge.net/

[17] Wireshark. http://www.wireshark.org/

[18] Regshot. http://sourceforge.net/projects/regshot/

[19] Norman Sandbox. http://sandbox.norman.no

[20] Willems, C., Holz, T. and Freiling, F. (2007) Toward Automated Dynamic Malware Analysis Using Cwsandbox. *IEEE Security & Privacy*, **5**, 32-39. http://dx.doi.org/10.1109/MSP.2007.45

[21] Anubis. http://anubis.iseclab.org/

[22] Bayer, U., Kruegel, C. and Kirda, E. (2006) TTAnalyze: A Tool for Analyzing Malware. *Proceedings of the* 15*th European Institute for Computer Antivirus Research Annual Conference*.

[23] Dinaburg, A., Royal, P., Sharif, M. and Lee, W. (2008) Ether: Malware Analysis via Hardware Virtualization Extensions. *Proceedings of the* 15*th ACM Conference on Computer and Communications Security*, *CCS*'08, Alexandria, 27-31 October 2008, 51-62.

[24] ThreatExpert. http://www.threatexpert.com/submit.aspx

[25] Schultz, M., Eskin, E., Zadok, F. and Stolfo, S. (2001) Data Mining Methods for Detection of New Malicious Executables. *Proceedings of* 2001 *IEEE Symposium on Security and Privacy*, Oakland, 14-16 May 2001, 38-49.

[26] Cohen, W. (1995) Fast Effective Rule Induction. *Proceedings of* 12*th International Conference on Machine Learning*, San Francisco, 115-123.

[27] Kolter, J. and Maloof, M. (2004) Learning to Detect Malicious Executables in the Wild. *Proceedings of the* 10*th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 470-478.

[28] Nataraj, L., Karthikeyan, S., Jacob, G. and Manjunath, B. (2011) Malware Images: Visualization and Automatic Classification. *Proceedings of the* 8*th International Symposium on Visualization for Cyber Security*, Article No. 4.

[29] Nataraj, L., Yegneswaran, V., Porras, P. and Zhang, J. (2011) A Comparative Assessment of Malware Classification Using Binary Texture Analysis and Dynamic Analysis. *Proceedings of the* 4*th ACM Workshop on Security and Artificial Intelligence*, 21-30.

[30] Kong, D. and Yan, G. (2013) Discriminant Malware Distance Learning on Structural Information for Automated Malware Classification. *Proceedings of the ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems*, 347-348.

[31] Tian, R., Batten, L. and Versteeg, S. (2008) Function Length as a Tool for Malware Classification. *Proceedings of the* 3*rd International Conference on Malicious and Unwanted Software*, Fairfax, 7-8 October 2008, 57-64.

[32] Tian, R., Batten, L., Islam, R. and Versteeg, S. (2009) An Automated Classification System Based on the Strings of Trojan and Virus Families. *Proceedings of the* 4*th International Conference on Malicious and Unwanted Software*, Montréal, 13-14 October 2009, 23-30.

[33] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I. (2009) The WEKA Data Mining Software: An Update. *ACM SIGKDD Explorations Newsletter*, 10-18.

[34] Santos, I., Nieves, J. and Bringas, P.G. (2011) Semi-Supervised Learning for Unknown Malware Detection. *International Symposium on Distributed Computing and Artificial Intelligence Advances in Intelligent and Soft Computing*, **91**, 415-422.

[35] Moskovitch, R., Stopel, D., Feher, C., Nissim, N. and Elovici, Y. (2008) Unknown Malcode Detection via Text Categorization and the Imbalance Problem. *Proceedings of the* 6*th IEEE International Conference on Intelligence and Security Informatics*, Taipei, 17-20 June 2008, 156-161.

[36] Santos, I., Nieves, J. and Bringas, P.G. (2011) Collective Classification for Unknown Malware Detection. *Proceedings of the International Conference on Security and Cryptography*, Seville, 18-21 July 2011, 251-256.

[37] Siddiqui, M., Wang, M.C. and Lee, J. (2009) Detecting Internet Worms Using Data Mining Techniques. *Journal of Systemics, Cybernetics and Informatics*, **6**, 48-53.

[38] Zolkipli, M.F. and Jantan, A. (2011) An Approach for Malware Behavior Identification and Classification. *Proceeding of* 3rd *International Conference on Computer Research and Development*, Shanghai, 11-13 March 2011, 191-194.

[39] Rieck, K., Trinius, P., Willems, C. and Holz, T. (2011) Automatic Analysis of Malware Behavior Using Machine Learning. *Journal of Computer Security*, **19**, 639-668.

[40] Anderson, B., Quist, D., Neil, J., Storlie, C. and Lane, T. (2011) Graph Based Malware Detection Using Dynamic Analysis. *Journal in Computer Virology*, **7**, 247-258. http://dx.doi.org/10.1007/s11416-011-0152-x

[41] Bayer, U., Comparetti, P.M., Hlauschek, C. and Kruegel, C. (2009) Scalable, Behavior-Based Malware Clustering. *Proceedings of the* 16*th Annual Network and Distributed System Security Symposium*.

[42] Indyk, P. and Motwani, R. (1998) Approximate Nearest Neighbor: Towards Removing the Curse of Dimensionality. *Proceedings of* 30*th Annual ACM Symposium on Theory of Computing*, Dallas, 24-26 May 1998, 604-613.

[43] Tian, R., Islam, M.R., Batten, L. and Versteeg, S. (2010) Differentiating Malware from Cleanwares Using Behavioral Analysis. *Proceedings of* 5*th International Conference on Malicious and Unwanted Software* (*Malware*), Nancy, 19-20 October 2010, 23-30.

[44] Biley, M., Oberheid, J., Andersen, J., Morley Mao, Z., Jahanian, F. and Nazario, J. (2007) Automated Classification and Analysis of Internet Malware. *Proceedings of the* 10*th International Conference on Recent Advances in Intrusion Detection*, **4637**, 178-197. http://dx.doi.org/10.1007/978-3-540-74320-0_10

[45] Park, Y., Reeves, D., Mulukutla, V. and Sundaravel, B. (2010) Fast Malware Classification by Automated Behavioral Graph Matching. *Proceedings of the* 6*th Annual Workshop on Cyber Security and Information Intelligence Research*, Article No. 45.

[46] Firdausi, I., Lim, C. and Erwin, A. (2010) Analysis of Machine Learning Techniques Used in Behavior Based Malware Detection. *Proceedings of* 2*nd International Conference on Advances in Computing*, *Control and Telecommunication Technologies* (*ACT*), Jakarta, 2-3 December 2010, 201-203.

[47] Nari, S. and Ghorbani, A. (2013) Automated Malware Classification Based on Network Behavior. *Proceedings of International Conference on Computing*, *Networking and Communications* (*ICNC*), San Diego, 28-31 January 2013, 642-647.

[48] Lee, T. and Mody, J.J. (2006) Behavioral Classification. *Proceedings of the European Institute for Computer Antivirus Research Conference* (EICAR'06).

[49] Santos, I., Devesa, J., Brezo, F., Nieves, J. and Bringas, P.G. (2013) OPEM: A Static-Dynamic Approach for Machine Learning Based Malware Detection. *Proceedings of International Conference CISIS*'12-*ICEUTE*'12, Special Sessions *Advances in Intelligent Systems and Computing*, **189**, 271-280.

[50] Islam, R., Tian, R., Battenb, L. and Versteeg, S. (2013) Classification of Malware Based on Integrated Static and Dynamic Features. *Journal of Network and Computer Application*, **36**, 646-556. http://dx.doi.org/10.1016/j.jnca.2012.10.004

[51] Anderson, B., Storlie, C. and Lane, T. (2012) Improving Malware Classification: Bridging the Static/Dynamic Gap. *Proceedings of* 5*th ACM Workshop on Security and Artificial Intelligence* (*AISec*), 3-14.