American Journal of Applied Sciences 9 (3): 283-288, 2012
ISSN 1546-9239

Malware Detection Based on Hybrid Signature Behaviour Application Programming Interface Call Graph

Ammar Ahmed E. Elhadi, Mohd Aizaini Maarof and Ahmed Hamza Osman
Information Assurance and Security Research Group,
Faculty of Computer Science and Information Systems,
University Technology, Malaysia

Abstract: Problem statement: A malware is a program that has malicious intent. Nowadays, malware authors apply several sophisticated techniques such as packing and obfuscation to avoid malware detection. That makes zero-day attacks and false positives the most challenging problems in the malware detection field. **Approach:** In this study, the static and dynamic analysis techniques that are used in malware detection are surveyed. Static analysis techniques, dynamic analysis techniques and their combination including Signature-Based and Behaviour-Based techniques are discussed. **Results:** In addition, a new malware detection framework is proposed. **Conclusion:** The proposed framework combines Signature-Based with Behaviour-Based using API graph system. The goal of the proposed framework is to improve accuracy and scan process time for malware detection.

Key words: Malware detection, API call graph, framework

INTRODUCTION

Malware stands for malicious software. It is the type of software that is designed with a harmful intent in mind. It comes in many forms such as Viruses, Worms, Trojan horses, Backdoors, Spyware, Rootkits, botnet in addition to other types of software with unwanted behavior (Wang, 2006).

The following are breif descriptions for each of the above mentioned malware types (Wang, 2006):

- Viruses are programs that self-replicate within a host by attaching themselves to programs and/or documents that become carriers of the malicious code
- Worms are programs that self-replicate across a network
- Trojan horses masquerade as useful programs, but contain malicious code to attack the system or leak data
- Back doors open the system to external entities by subverting the local security policies to allow remote access and control over a network
- Spyware is a useful software package that also transmits private user data to an external entity
- Rootkits is a collection of tools often used by an attacker after gaining administrative privileges on a host
- Botnet is remotely controlled software that comprises a collection of autonomous software tools
- Malware detector is a system that attempts to identify malware using signatures and other heuristics techniques; Antivirus scanner is an example of a malware detector (Wang, 2006); the malware writer (hacker) on the other hand applies sophisticated techniques to evade detection by modifying or morphing malware using packing techniques and/or program obfuscation. Two common obfuscation techniques are Polymorphism and Metamorphism (You and Yim, 2010)

The malware detector attempts to help protect the system by detecting malicious behavior. The malware detector may or may not reside on the same system it is trying to protect. Malware detectors take two inputs:

- Knowledge of the malware signature or behavior (learning)
- The program under inspection

Once the malware detector has the knowledge of what is considered malware behavior (abnormal behavior) and the program under inspection, it can employ its detection technique to decide if the program is malware or benign.

Corresponding Author: Ammar Ahmed E. Elhadi, Information Assurance and Security Research Group,
Faculty of Computer Science and Information Systems, University Technology, 81310, Malaysia

MATERIALS AND METHODS

Malware analysis can be categorized into two main categories:

- Analysis of the infected file without executing it, which is known as static analysis. In this approach, we extract low-level information such as Control Flow Graphs (CFGs), Data-Flow Graphs (DFGs) and System call analysis. This information can be gathered by disassembling or decompiling the infected file using tools like IDA Pro (Riesen and Bunke, 2009). Sometimes analyzing the infected file in a different environment to avoid auto execution of the malware is better. Using static analysis we get fast, safe and low false positives and we trace all paths, which helps in terms of getting a lot of information to analyze. On the other hand static analysis may fail in analyzing unknown malware that uses code obfuscation techniques (Egele *et al*., 2011)
- Analysis of the infected file during its execution, which is known as dynamic analysis. Dynamic analysis executes the infected file on simulated environment (a debugger or a virtual machine or an emulator) to analyze its malicious functions. The analysis environment must be invisible to the malware because the malware writer use tools like anti-virtual machine and Anti-emulation to hide their malware functions if they detect they are under analysis. Dynamic analysis fails to detect activities of interest if the target changes its behavior depending on trigger conditions such as existence of a specific file or specific day as only a single execution path may be examined for each attempt (Egele *et al*., 2011)

Techniques: There are mainly two techniques for malware detection: Signature-Based and Behavior-Based techniques (Table 1-2).

In signature-based techniques a sequence of instructions unique to a malware is used to generate a malware signature, which is captured by researchers in a laboratory environment (Goertzel, 2009). A signature should be able to identify any malware exhibiting the malicious behavior specified by the signature. Most of antivirus scanners are signature based.

Behavior-based detection techniques focus on analyzing the behavior of known and suspected malicious code. Such behaviors include factors such as the source and destination addresses of the malware, the attachment types in which they are embedded and statistical anomalies in malware infected systems (Goertzel, 2009). One example of a behavior-based detection approach is the histogram-based malicious code detection technology patented by Symantec.

To overcome the limitations of signature-based detection some malware researchers apply graph (Control Flow Graph (Bonfante *et al*., 2007), Call graph (Lee *et al*., 2010), machine learning techniques (Rieck, *et al*., 2011) and data mining techniques (Kephart and Arnold, 1994; Schultz *et al*., 2001) (Objective-Oriented Association (OOA) (Ye *et al*., 2008)).

Other researchers apply techniques like finite automaton, HMM, data mining (Schultz *et al*., 2001; Siddiqui *et al*., 2008) (Naïve Bayes, Support Vector Machine (SVM) and Decision tree) and neural network (Tesauro *et al*., 1996) to improve Behavior-Based detection.

Related work: Malware Detection is divided into two methods: Signature-Based and Behavior-Based techniques and each technique can be applied using static analysis or dynamic analysis or hybrid analysis (Idika and Mathur, 2007), Fig. 1-3 shows the organization of malware detection.

Implementing signature based detection without executing the suspected file (Static analysis) was the first try to detect malware. Researchers applied different techniques to improve detection rate. Some of them applied Objective-Oriented Association (OOA) mining based classification (Ye *et al*., 2008). Their model consisted of three major modules: PE parser, OOA rule generator and rule based classifier. After a while they developed their work using postprocessing techniques associative classification method based on the analysis of Application Programming Interface (API) execution calls (Ye *et al*., 2010). Other researchers combined signature-based technique and genetic algorithm technique, but their study focused on three types of malware which are virus, worms and Trojan horse (Zolkipli and Jantan, 2010).

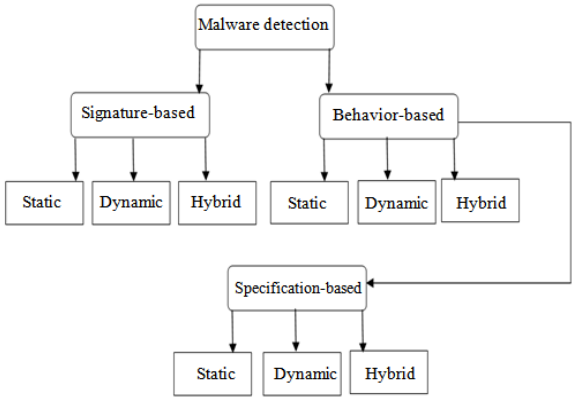

Fig. 1: Organization of malware detection

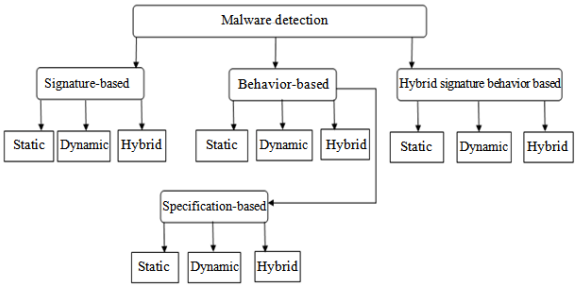

Fig. 2: New organization of malware detection

Table 1: Summary of the advantages and disadvantage of static and dynamic analysis

	Advantage	Disadvantage
Static analysis	Fast and safe. Low level of false positives. Good in analyzing multipath malware.	Difficulty analyzing unknown malware.
Dynamic analysis	Good in detecting unknown malware.	Neither fast nor safe. Difficulty analyzing multipath malware.

Table 2: Summary of the advantages and disadvantages of signature-based and behavior-based techniques

	Advantage	Disadvantage
Signature-based	Less scanning time.	Unknown malware can easily evade detection.
	Few false positives.	Cannot deal with simple obfuscation.
Behavior-based	Best results in detecting of polymorphic malware.	Not able to detect a lot of polymorphic viruses present (Packers).

Signature based detection was also applied during suspected file execution (dynamic analysis) in which the researchers trace API calls and then build their suspected file signature (Nair *et al*., 2010), this researcher generated signature for an entire malware class instead of for individual malware samples. Once a base signature for a particular metamorphic generator is generated, all the metamorphic viruses created from that tool are easily detected.

Most of the existing works relies on using behavior based detection where some researchers apply static analysis while others apply dynamic analysis. Some of the works focus on kernel memory mapping to develop a malware behavior monitor that uses a temporal view of kernel objects in the analysis of kernel execution traces (Rhee *et al*., 2010). Other focus on avoiding false positives by tracing malware behavior usually not do but installers and uninstallers do (Fukushimayz *et al*., 2010; (Park *et al*., 2010) propose a new malware classification method based on maximal common subgraph detection.

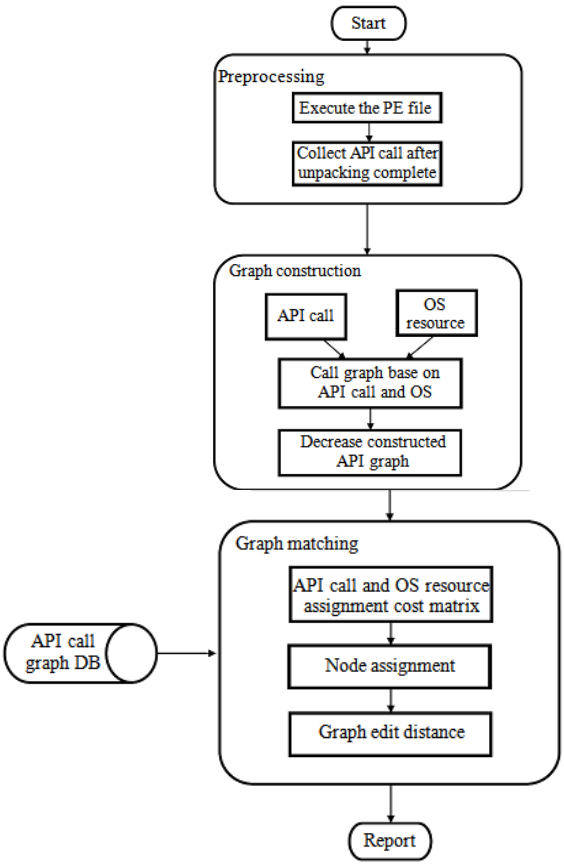

Fig. 3: Proposed framework

Current researchers combine static analysis with dynamic analysis to overcome the limitations of each method. Guo *et al*. (2010) proposed a framework that combined static and dynamic binary translation features to detect malware and prevent its execution. They apply behavior Control Flow Graph (CFG) and then critical API Graph based on CFG is generated to do sub-graph matching. Other researchers apply signature Control Flow Graph (CFG) and use edit distance matches between graphs.

As demonstrated in the previous paragraph the observation is that now some malware researchers focus on graph (control flow graph, call graph, code graph). They build their graph in different ways and analyze and compare graph using different methods. To build the graph most researchers present node graph as system call. For example (Lee *et al*., 2010) creates their graph by transforming PE file into call graph, the call graph nodes are system calls and the edges are system call sequence. Then the call graph minimize into code graph to speed up the analyze and compare graphs. Other researchers (Park *et al*., 2010) use the same way

by use 4-tuples node corresponds system call, edges the dependency of two system calls and label for nodes and edges. Some other researchers define node graph as kernel objects rather than system calls (Park and Reeves, 2011). On the other hand (Kostakis *et al*., 2011) built the graph from the subroutines as nodes and their call references as edges, (Kim and Moon, 2010) they use a dependency graph whose vertex represents a line in the semantic code. The dependency between two lines is represented by a directed edge and (Bai *et al*., 2009; Guo *et al*., 2010) extract a Critical API Graph (CAG) from a Control Flow Graph (CFG) for each malware to define the behavior.

RESULTS AND DISCUSSION

The above works compare graph using different graph matching techniques some of them use maximal common subgraph (Kim and Moon, 2010; Park and Reeves, 2011) and some use Weighted Common Behavioral Graph Generation based on an Approximate Algorithm and others build formula using intersection and the union of the graphs (Lee *et al*., 2010) but all require time and space due to NP-completeness of the problem.

Proposed framework: Since each technique has advantages and disadvantages, it is believed that by combining them in some manner we can improve the advantages and decrease the disadvantages. Using static techniques we can get fast and safe result and also by applying unpacking tools to solve packing problem and analysis of the file using both signature and behavior-based methods we can get better results. In case static techniques fail, we can use dynamic techniques to do more analysis on the file. Furthermore, to get more efficient result about the infected file, we can analyze the file using both signature and behavior-based methods.

Framework component:

Execute the PE file and collect API calls: Execute the suspected file in safe (apply rootkit) and controlled environment and use kernel hooking to extract API call after unpacking if the file is packed. We will to trace different path in the file.

Construct the hybrid call graph: We build our graph using API call collecting from the execution of the file; our graph differs from other researcher's graphs in that we build it from the API calls and the operating system resources used by API call as graph nodes, the edges represent the reference between the nodes. Our nodes will have two attributes: API call and operating system resource, the graph label is the API calls its self or the operating system resource.

The construction of the API call graph for a program without API operating system recourses is very simple. In programs containing API operating system recourses, it is possible to have a reference to API operating system recourses which may represent invocations of several distinct other API calls. In order to address all possible questions which result from such a references in API call, we need to know all other API calls associated with that API operating system recourses (Ryder, 1979).

Decrease the constructed graph: The generated graph from the previous step contains huge number of nodes and edges and needs to be minimized. This operation will be by removing unused instruction (junk code, computation) and focus on popular API call used by the majority of malware.

We can use the information on node (API call, operating system resources) to build our API call graph database.

Finding matching graphs: Graph Edit Distance (GED) is the best algorithm for matching inexact graph type (Gao *et al*., 2010; Riesen *et al*., 2010) but its complexity makes it slow (Riesen and Bunke, 2009). To speed up GED we need to find an assignment between the nodes of the two compared graph. For assignment problem we need to build API call and operating system resource cost matrix from the two compared API graphs, after that we can apply an assignment algorithm (Munkres' algorithm) (Munkres, 1957; Riesen and Bunke, 2009) to assign node form one graph to other graph with minimal cost . One difficult in GED using an assignment algorithm (Munkres' algorithm) its base on minimum cost matrix for API call and operating system resources node and edges, where it assumed the cost is fixed value between them (He and Singh, 2006), to minimize the cost matrix, more near nodes and edges are to matching (Hu *et al*., 2009).

Hu *et al*. (2009) they develop modified Hungarian algorithm based on neighbor matching. Our call graph based on structure and attribute graph, to minimize the cost of the cost matrix we will partition the data graph into sub-graphs based on structure connectivity and attribute connectivity (Zhu *et al*., 2011).

CONCLUSION

In this study we have shown that signature based techniques and behavior based techniques can be combine to build a system that has better detection of

polymorphic malware and less time scan. We have proposed new framework using API call graph system to implement this combination and we have built the system using dynamic analysis method.

ACKNOWLEDGMENT

The researchers would like to thanks University Technology Malaysia for the unlimited support. And the significant role of Sudanese Research Community in UTM is highly appreciated.

REFERENCES

Bai, L., J. Pang, Y. Zhang, W. Fu and J. Zhu, 2009. Detecting malicious behavior using critical API-calling graph matching. Proceedings of the 1st International Conference on Information Science and Engineering, Dec. 26-28, IEEE Xplore Press, Nanjing, pp: 1716-1719. DOI: 10.1109/ICISE.2009.494

Bonfante, G., M. Kaczmarek and J.Y. Marion, 2007. Control flow graphs as malware signatures. Nancy-Universite-Loria. http://hal.archives-ouvertes.fr

Egele, M., T. Scholte, E. Kirda and C. Kruegel, 2011. A survey on automated dynamic malware analysis techniques and tools. ACM Comput. Surv., 5: 1-49.

Fukushimayz, Y., A. Sakai, Y. Horiyz and K. Sakuraiyz, 2010. A behavior based malware detection scheme for avoiding false positive. Proceedings of the 6th IEEE Workshop on Secure Network Protocols, Oct 5-5, IEEE Xplore Press, Kyoto, pp: 79-84. DOI: 10.1109/NPSEC.2010.5634444

Gao, X., B. Xiao, D. Tao and X. Li, 2010. A survey of graph edit distance. Patt. Anal. Appli., 13: 113-129. DOI: 10.1007/s10044-008-0141-y

Goertzel, K.M., 2009. Tools on Anti Malware. Technical Information Center.

Guo, H., J. Pang, Y. Zhang, F. Yue and R. Zhao, 2010. HERO: A novel malware detection framework based on binary translation. Proceedings of the IEEE International Conference on Intelligent Computing and Intelligent Systems, Oct. 29-31, IEEE Xplore Press, Xiamen, pp: 411-415. DOI: 10.1109/ICICISYS.2010.5658586

He, H. and A.K. Singh, 2006. Closure-tree: An index structure for graph queries. Proceedings of the 22nd International Conference on Data Engineering, Apr. 03-07, IEEE Xplore Press, pp: 38-38. DOI: 10.1109/ICDE.2006.37

Hu, X., T.C. Chiueh and K.G. Shin, 2009. Large-scale malware indexing using function-call graphs. Proceedings of the 16th ACM Conference on Computer and Communications Security, Nov. 09-13, ACM, USA., pp: 611-620. DOI: 10.1145/1653662.1653736

Idika, N. and A.P. Mathur, 2007. A survey of malware detection techniques. Purdue University. http://scholar.googleusercontent.com

Kephart, J.O. and W.C. Arnold, 1994. Automatic extraction of computer virus signatures. Proceedings of the 4th Virus Bulletin International Conference, (VBIC'94), Virus Bulletin Ltd., Abingdon, England, pp: 178-184.

Kim, K. and B.R. Moon, 2010. Malware detection based on dependency graph using hybrid genetic algorithm. Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, Jul. 07-11, ACM, USA., pp: 1211-1218. DOI: 10.1145/1830483.1830703

Kostakis, O., J. Kinable, H. Mahmoudi and K. Mustonen, 2011. Improved call graph comparison using simulated annealing. Proceedings of the 2011 ACM Symposium on Applied Computing, Mar. 21-24, ACM, USA., pp: 1516-1523. DOI: 10.1145/1982185.1982509

Lee, J., K. Jeong and H. Lee, 2010. Detecting metamorphic malwares using code graphs. Proceedings of the 2010 ACM Symposium on Applied Computing, Mar. 22-26, ACM, USA., pp: 1970-1977. DOI: 10.1145/1774088.1774505

Munkres, J., 1957. Algorithms for the assignment and transportation problems. J. Soc. Indus. Applied Math., 5: 32-38.

Nair, V.P., H. Jain, Y.K. Golecha, M.S. Gaur and V. linkLaxmi, 2010. MEDUSA: MEtamorphic malware dynamic analysis usingsignature from API. Proceedings of the 3rd International Conference on Security of Information and Networks, Sep. 07-11, ACM, USA., pp: 263-269. DOI: 10.1145/1854099.1854152

Park, Y., D. Reeves, V. Mulukutla and B. Sundaravel, 2010. Fast malware classification by automated behavioral graph matching. Proceedings of the 6th Annual Workshop on Cyber Security and Information Intelligence Research, Apr. 21-23, ACM, USA., DOI: 10.1145/1852666.1852716

Park, Y. and D. Reeves, 2011. Deriving common malware behavior through graph clustering. Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, Mar. 22-24, ACM, USA., pp: 497-502. DOI: 10.1145/1966913.1966986

Rhee, J., R. Riley, D. Xu and X. Jiang, 2010. Kernel malware analysis with un-tampered and temporal views of dynamic kernel memory. Recent Adv. Intrusion Detect., 6307: 178-197. DOI: 10.1007/978-3-642-15512-3_10

Rieck, K., P. Trinius, C. Willems and T. Holz, 2011. Automatic analysis of malware behavior using machine learning. J. Comput. Security, 19: 639-668.

Riesen, K. and H. Bunke, 2009. Approximate graph edit distance computation by means of bipartite graph matching. Image Vision Comput., 27: 950-959. DOI: 10.1016/j.imavis.2008.04.004

Riesen, K., X. Jiang and H. Bunke, 2010. Exact and inexact graph matching: Methodology and applications. Manag. Min. Graph Data, 40: 217-247. DOI: 10.1007/978-1-4419-6045-0_7

Ryder, B.G., 1979. Constructing the call graph of a program. IEEE Trans. Software Eng., 5: 216-226. DOI: 10.1109/TSE.1979.234183

Schultz, M.G., E. Eskin, F. Zadok and S.J. Stolfo, 2001. Data mining methods for detection of new malicious executables. Proceedings of the IEEE Symposium on Security and Privacy, May 14-16, IEEE Xplore Press, Oakland, CA, USA., pp: 38-49. 10.1109/SECPRI.2001.924286

Siddiqui, M., M.C. Wang and J. Lee, 2008. A survey of data mining techniques for malware detection using file features. Proceedings of the 46th Annual Southeast Regional Conference on XX, Mar. 28-28, ACM, USA., pp: 509-510. DOI: 10.1145/1593105.1593239

Tesauro, G.J., J.O. Kephart and G.B. Sorkin *et al*., 1996. Neural networks for computer virus recognition. IEEE Expert, 11: 5-6. DOI: 10.1109/64.511768

Wang, C., 2006. Malware Detection. 1st Edn., Springer, New York, ISBN: 0387327207, pp: 311.

Ye, Y., D. Wang, T. Li, D. Ye and Q. Jiang, 2008. An intelligent PE-malware detection system based on association mining. J. Comput. Virol., 4: 323-334. DOI: 10.1007/s11416-008-0082-4

Ye, Y., T. Li, Q. Jiang and Y. Wang, 2010. CIMDS: Adapting postprocessing techniques of associative classification for malware detection. IEEE Trans. Syst. Man Cybernetics Part C: Appli. Rev., 40: 298-307. DOI: 10.1109/TSMCC.2009.2037978

You, I. and K. Yim, 2010. Malware obfuscation techniques: A brief survey. Proceedings of the International Conference on Broadband, Wireless Computing, Communication and Applications, Nov. 4-6, IEEE Xplore Press, Fukuoka, pp: 297-300. DOI: 10.1109/BWCCA.2010.85

Zhu, L., W.K. Ng and J. Cheng, 2011. Structure and attribute index for approximate graph matching in large graphs. Inform. Syst., 36: 958-972. DOI: 10.1016/j.is.2011.03.009

Zolkipli, M.F. and A. Jantan, 2010. A framework for malware detection using combination technique and signature generation. Proceedings of the 2nd International Conference on Computer Research and Development, May 7-10, IEEE Xplore Press, Kuala Lumpur, pp: 196-199. DOI: 10.1109/ICCRD.2010.25