
Static Heuristics Classifiers as Pre-Filter for Malware Target Recognition (MATR)

Anuj Lohani, Aditi Lohani, Jitendra Singh, Manish Bhardwaj

Dept. of Computer Science and Engineering, SRM University, NCR Campus, Modinagar, India

Email address:

lo.juna@gmail.com (A. Lohani), aditilohani@gmail.com (A. Lohani), jitendra.s@ncr.srmuniv.ac.in (J. Singh),

aapkaapna13@gmail.com (M. Bhardwaj)

To cite this article:

Anuj Lohani, Aditi Lohani, Jitendra Singh, Manish Bhardwaj. Static Heuristics Classifiers as Pre-Filter for Malware Target Recognition (MATR). *American Journal of Networks and Communications*. Vol. 4, No. 3, 2015, pp. 44-48. doi: 10.11648/j.ajnc.20150403.14

Abstract: Now a day's malware are one of the major threats to computer information system. The current malware detection technologies have certain significant limitations on their part. Different organizations which deal with the protection of sensitive information may face the problem in identifying recent malware threats among millions and billions of benign executables using just signature-based antivirus systems. Currently for frontline defense against malware, signature-based antivirus products are used by organization. In the undergoing project, we proposed a detection approach by using static heuristics in MATR for malware in PE (portable executable) files. The project suggests larger performance-based malware target recognition architecture that at present use only static heuristic features. Results of the experiments show that this architecture achieves an overall test accuracy of greater than 98% against malware set collected from various operational environments, while most antivirus provide detection accuracy of only 60% at their most sensitive configuration [1]. Implementations of this architecture enables benign executables to be classified successfully to some extent providing enhanced awareness of operators in hostile environments it also enable detection of unknown malware. We are to show the performance of Bagging and AdaBoost ensemble.

Keywords: Malware, PE (Portable Executable), Bagging, AdaBoost (Adaptive Boosting)

1. Introduction

Computer networks of large organizations are difficult to secure and faces challenges due to their vast scale, scope, and complexity. The analysts must go thoroughly through a potentially overwhelmed set of data to discover new malware threats. As the data sets fails to be reduced to manageable extent in large organizations, the attacker tools and malicious network traffic successfully hide in plain sight among millions of executable programs and billions of network connections making it difficult for humans to find without a form of automated assistance.

This field of malware detection has now been an active in computer security research area for decades. Advances in this area have not produced much significant solutions for this malware problem. The solutions must enhance human effectiveness to defend the network rather than replacing them.

Known threats are effectively detected and identified by signature-based antivirus and intrusion detection systems, but these could be nevertheless efficient with respect to new and

unknown threats. For persons attacking these systems work as design constraints while designing new tools in order to avoid detection via system. [18]

An appreciable and useful data reduction might not be performed by these systems to reduce workload of analysts as their detections are typical binary yes or no outputs.

The research on malware detection provides a potentially viable method of data reduction of analyst workload. The heuristic analysis techniques are categorized into two distinct categories: static and dynamic. Static heuristics employs non-runtime indicators, such as structural anomalies, n-grams and program disassembly while on the other hand dynamic heuristics uses runtime indicators obtained in virtual environments, such as commercial sandbox applications. [1]

Both of these techniques complement one another (and even commercial antivirus products) hence neither static nor dynamic analysis is sufficient alone for providing full spectrum defense, with reduced effective scan and detection time against malware. Theoretically, sensitive and relatively fast static analysis methods can serve as pre-filters for slower dynamic analysis methods thus reducing the effective runtime scan performance of the overall system while producing a

lower number of false positives than either method alone.[1] Contributions by this research are:

- 1) The MaTR architecture is extended which initially proposed, an operational model for self-discovery of malware in an organization with low effective scan times and low false positive rates through successive data reduction and analysis.
- 2) Provides generalization of the static analysis models trained from one or more current dataset.

These applications are challenging for research and commercial antivirus solutions, as they signify the prominent threats that were not consider prior for deployment by defensive system. Also the test environment considers various levels of sensitivity, simulation and organizational execution in unfriendly cyberspace environments to normal environment. [1]

2. Related Work

Recent static heuristic analysis research focuses on the occurrence in programs of n-grams, which are byte sequences of length. N-grams approach is applied for feature source which avoids the problems of generating pristine disassembly described by Moser [4], but non-instruction-based static heuristics like N-gram are not the only approach certainly available today. Researchers have also employed strings [2], [3], [5] and anomaly and structure data [2], [9], [6], [7] as features for malware detection classifiers. Presently, the scope of MaTR is strictly static heuristic analysis. The section describes related research work in static heuristic analysis of malware and detection evasion techniques. [1]

2.1. Static Approach

Static approach only checks executable binaries or assembly code without executing the codes. According to a survey done not long ago [9] the static approach attain a very high accuracy while maintaining low false positives while implying machine learning classifiers on static features in order to detect unknown malware. The advantage of this approach is that it does not add up overhead of execution time. [8]

Kolter [10] and Maloof improved Schulz's technique, by introducing byte n-grams rather than non-overlapping sequences. 500 n-grams are selected through measuring information gain and 4grams are employing as features. They applied decision trees; naïve instance based learners, support vector machines and TFIDF and, also improved some of these classification algorithms. Best result was achieved by boosted J48 at AUC, 0.996. [8]

2.2. N-Gram Features

The IBM research of Kephart, Tesauro, and Arnold provides the seminal research in n-gram analysis of malware [11]–[13]. These n-grams are byte sequences of length n that occur in the target, which theoretically represent program structural components and fragments of instructions and data.

They examine the use of n-grams in automatic signature extraction [11] for malware variants and generic detection [12], [13].

While searching for methods to automate signature extraction for new variants of known malware, Kephart et al. [11] discovered the utility of n-grams for generic malware detection. By determining the probability of finding specific n-grams in malicious and non-malicious programs, the authors fabricated a generic malware detection classifier.

Tesauro et al. [13] successfully used neural networks to detect boot sector viruses. They manipulated the decision threshold boundary to increase the cost associated with false positives as they cited that a single false positive reading likely affects thousands of systems. Despite significant computational and space constraints, and a small sample size for training and testing, they achieved a false positive rate (FPR) of less than 1% while detecting over 80% of unknown boot sector viruses.

3. Problem Description

Apart from performance issues MaTR has other issues to handle; these issues can be classified as problem other than performance heuristic based component. The section gives us the spread of problems, their possible solutions and the system process for MaTR.

3.1. Existing Problem

The problem within the current scenario of the PE's with malware and their detection can be described as follows:

- 1) Detection schemes which currently exist are based on an implicit assumption that each infected computer conducts scan over the Internet and propagate itself at the highest speed possible.
- 2) It has been demonstrated that the traffic volume of worm scan and the number of infected computers via worm show exponentially increasing patterns.
- 3) Worm may also make the use of traffic morphing and evasive scan technique to hide detection.
- 4) An analysis of a MaTR system presumes a decentralized deployment. A centralized deployment of a MaTR system makes it less vulnerable to adversary exfiltration and analysis.
- 5) The MaTR architecture may be susceptible to reverse engineering and exploitation.

3.2. Proposed Solution

The approachable solutions for the problem of malware detection that can be can be presented as follows:

- 1) Proposed Worm detection schemes are based on the global traffic monitor, scan and anomalous behavior in traffic. Other defense and worm detection methods such as sequential hypothesis testing applied for detecting worm-infected systems and payload-based worm signature detection are also present.
- 2) Use of a better performance-based static heuristic

classifier over MaTR architecture, to improve pre-filtering capability. Better detection of malware using both static and dynamic analysis, complementing each other.

- 3) A state-space feedback control model was presented which could control and detect spreading of worms or viruses. This is done by calculating the rate of new connections which the infected systems make.
- 4) In spite of different approaches mentioned above, we hold as an opinion that detection of wide scan anomalous behavior continues to be a productive means against worms, and practically multifaceted defense is advantageous.

3.3. MaTR Process

The MaTR system uses a simple and straight forward method for detecting malware by using only high-level structural data of a program. While many commercial companies and researchers use same structural data, none exclusively rely on this source of data and achieve the performance levels of MaTR. Fig. 1 shows the inputs and outputs of MaTR and illustrates its internal process. Inputs to MaTR are executable files, such as portable executable (PE) files common in the Microsoft Windows operating systems.

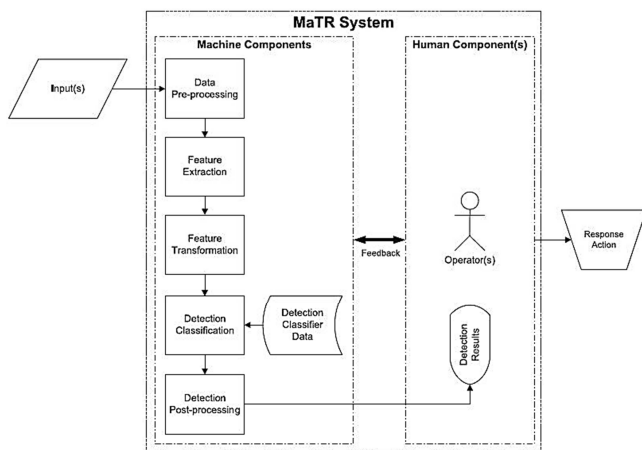


Figure 1. MaTR system process [16].

MaTR explicitly bounds the machine and human operator together within the overall system, a subtle yet significant distinction from other work that simply uses a computer to generate “solutions”. In MaTR’s architecture, the operator assertedly becomes a critical component receiving and providing feedback to the rest of the system and eventually initiating a response action. Since recognizing the operator’s role allows for a more robust network defense. The discoveries of certain malware payloads (an area of future work) require distinct response. [16]

4. Methodology

MaTR architecture works on prioritizing efforts on malware detection from an organization perspective. The network of

organizations can easily contain millions of programs, which is clearly infeasible to examine for a team of human analysts. Furthermore, if process behavior is observed using dynamic analysis for a 5 sec. period, it takes almost two months of time to record the behavior of a million processes, not including analysis time of the generated data. [1] The purpose of the MaTR architecture is to respond accurately to malware threats and enable organizations to efficiently discover threats. It accepts program files as input, after which the system makes predictions that enhance the effectiveness of malware analysts within the organization.

4.1. Static Analysis Component

The static component mostly targets malware defenses that result in program structural anomalies when compared to non-malware.

This component follows a standard machine learning process for malware detection using only non-instruction based static heuristics.

1) AdaboostMeta Classifier: Here static analysis component applies Adaboost meta classifier based on its performance on pilot studies, which can be observed as parallel findings of various researchers based on similar data sources. Boosting is a very different method to generate multiple predictions (function estimates) and combine them linearly. Boosting is a bias reduction technique, in contrast to bagging.

Boosting typically improves the performance of a single tree model. Reason for this is that we often cannot construct trees which are sufficiently large due to thinning out of observations in the terminal nodes. Boosting is then a device to come up with a more complex solution by taking linear combination of trees. Boosting is also very useful as a regularization technique for additive or interaction modeling.

4.2. Feature Set

A main difference between MaTR component and other commercial and research products are its feature set. MaTR component itself utilizes over 100 static heuristic features based on structural anomalies and structural information [1]. The feature set component in MaTR achieves high detection performance despite restricting its features exclusively to non-instruction based static heuristics. Other than following a mathematical model which are used to determine features, it utilizes anomaly and structural heuristic features commonly used by analysts [2], [6], [7] when examining samples to determine if they are indeed malicious.

To determine final feature set, MaTR component does not confide on complex computations of large samplings, thus it avert the selection steps overhead of a resource intensive feature. [1]

4.3. Experimental Details

The experiment uses a data sample of about 40137 malware, 10035 non-malware and 204 unknown sample instances. The data sample for malware came from VX Heavens dataset [17]

and data sample for non-malware sample came from windows clean installs. The sample was remodelled to numeric value according to the experiment. The tool for the analysis was Weka 3.7 tool, mainly used for data mining.

The ROC curve generated during the classification process depicts a relationship between true positive and false positive rate. The generalised role of classifiers in MaTRis to reduce the false positive rate, and to increase true positive rate of malware detection. The two cardinal classifier used for the comparison was AdaBoost and Bagging using decision tree J48 as classifier.

5. Result and Discussion

In this experiment after analysis from various sources AdaBoost was used against bagged decision tree classifier, as boosting algorithm are also very efficient and work well with decision tree. The test was performed with both Bagging and AdaBoost using J48 decision tree algorithm. The plot for ROC (receiver operating characteristic) curve was almost the same for both ensemble classifier, but while classifying the malware the relative absolute error and time to test model both was lower for AdaBoost.

First, we classified malware instance in the sample for both classifiers. Although samples were widely different, but for some unknown instance in sample the false positive rate for both classifiers was very low and the mean absolute error rate was also low considering the low percentage of correctly classified instances. The ROC for unknown instance this case was 0.9534. Some result is shown in Table 1.

Table 1. Initial test result.

	Bagging	AdaBoost
Mean absolute error	0.2241	0.224
False-positive	0.237	0.237
Root mean square error	0.3347	0.3347

Then, we classified both malware and non-malware sample for both classifier. Results were almost similar in both cases.

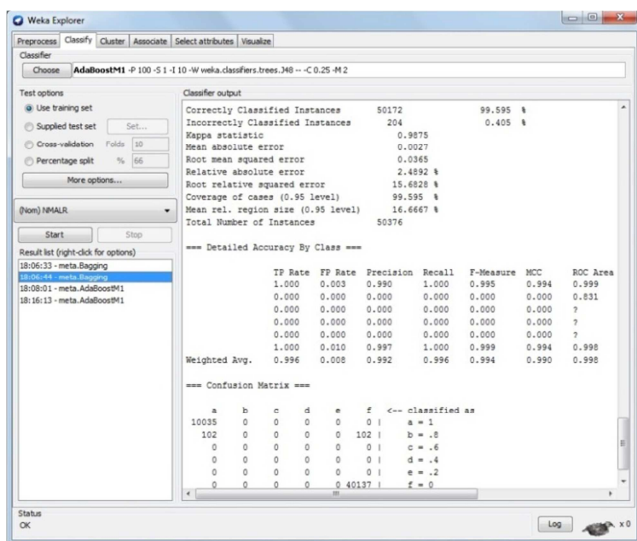


Figure 2. Test result using Bagging on malware and non-malware.

The Fig. 2 shows the test result for the bagging decision tree classifier in which the false positive rate was 0.008 and the value for the ROC curve was 0.998. The precision for the analysis was high at about 0.992 and relative absolute error was very low even for a large number of test instances.

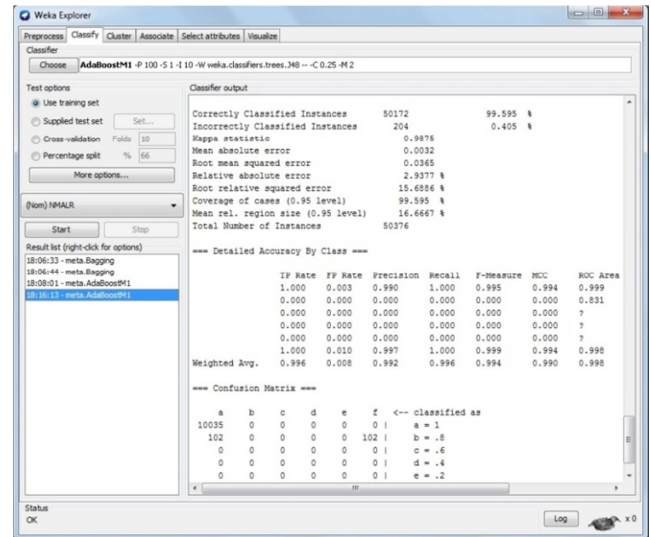


Figure 3. Test result using AdaBoost on malware and non-malware.

The test result for Fig. 3 shows that for the AdaBoost classifier in which the false positive rate was 0.008 and the value for the ROC curve was 0.998. The precision for the analysis was also high at 0.992 and relative absolute error was very low for a large number of test instances, even for this classifier.

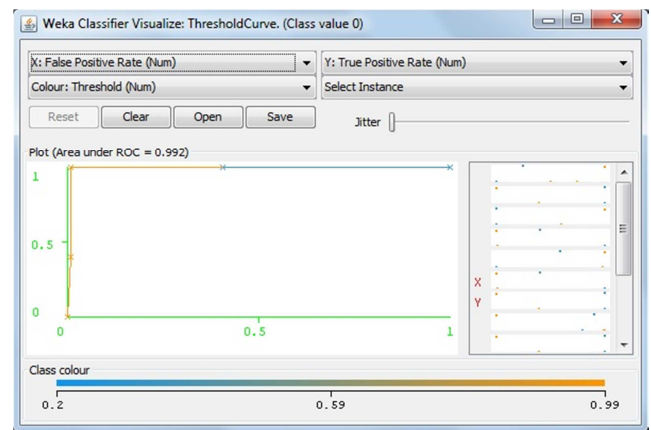


Figure 4. ROC curve using AdaBoost.

The Fig. 4 shows ROC curve using AdaBoost for one of the scenario. The result for the classification of malware and non-malware in case of both Bagging and AdaBoost classification provided us with almost similar result, though in most cases generated test result in less time and with some ups and downs for other stats in both cases.

6. Conclusion and Future Work

We have proposed approach by using AdaBoost as static

heuristic classifier for MaTR. This approach happens to maintain precise detection of unknown malware, based on samples that are inspected earlier, while also maintaining low false positive rate. The indications of experimental results are that, the precision of the classification algorithms is 0.990 and above, and theoretically applying multi-level ensemble algorithms may improve classification accuracy.

6.1. Conclusion

As already seen in the experiments the ensemble classifier AdaBoost provide more or less same performance as that of Bagging Meta classifier with J48 in detecting malware with 0.45% more relative absolute error. Thoughttime to test model for AdaBoost was less in most comparison, but when weclassified only malware the relative absolute error and time to test model respectively 0.02% and 1 sec lessforAdaBoost. Thus operational capability and accuracy was almost identical for both classifiers with some ups and downs for each.

6.2. Future Work

Examining performance characteristics of dynamic analysis methods to verify effective improvement in malware detection performance and analyze human-machine interface specifications. Given the capability of the MaTR architecture to provide focused information to the operator, generally accepted analysis processes may include redundant processes which humans can eliminate to increase overall performance further. Future investigations can also expand testing of other static heuristics classifiers as pre-filters for the MaTR architecture.

References

- [1] T. E. Dube, R. A. Raines, M. R. Grimaila, K. W. Bauer, S. K. Rogers, "Malware Target Recognition of Unknown Threats," *IEEE Systems Journal*, 2013.
- [2] P. Szor, "The Art of Computer Virus Research and Defense", IN: Addison-Wesley, 2005.
- [3] M. Schultz, E. Eskin, E. Zadok, and S. Stolfo, "Data mining methods for detection of new malicious executables," in *Proc. IEEE Symp. Security Privacy*, May 2001, pp. 38–49.
- [4] A. Moser, C. Kruegel, and E. Kirda, "Limits of static analysis for malware detection," in *Proc. ACSAC*, 2007, pp. 421–430.
- [5] M. Christodorescu, N. Kidd, and W.-H. Goh, "String analysis for x86 binaries," *ACM SIGSOFT Softw. Eng. Notes*, vol. 31, no. 1, p. 95, 2006.
- [6] N. Rafiq and Y. Mao, "Improving heuristics," *Virus Bull.*, pp. 9–12, Aug. 2008.
- [7] S. Treadwell and M. Zhou, "A heuristic approach for detection of obfuscated malware," in *Proc. Intell. Security Inform.*, Jun. 2009, pp. 291–299.
- [8] Jinrong Bai, Junfeng Wang, and Guozhong Zou, "A Malware Detection Scheme Based on Mining Format Information," *The Scientific World Journal Volume 2014*, Article ID 260905, 11 pages.
- [9] A. Shabtai, R. Moskovitch, Y. Elovici, and C. Glezer, "Detection of malicious code by applying machine learning classifiers on static features: a state-of-the-art survey," *Information Security Technical Report*, vol. 14, no. 1, pp. 16–29, 2009.
- [10] J. Z. Kolter and M. A. Maloof, "Learning to detect and classify malicious executables in the wild," *Journal of Machine Learning Research*, vol. 7, pp. 2721–2744, 2006.
- [11] J. O. Kephart and B. Arnold, "Automatic extraction of computer virus signatures," in *Proc. 4th Virus Bull. Int. Conf.*, 1994, pp. 178–184.
- [12] W. Arnold and G. Tesauro, "Automatically generated Win32 heuristic virus detection," in *Proc. Virus Bull. Conf.*, Sep. 2000, pp. 51–60.
- [13] G. Tesauro, J. Kephart, and G. Sorkin, "Neural networks for computer virus recognition," *IEEE Expert*, vol. 11, no. 4, pp. 5–6, Aug. 1996.
- [14] T. E. Dube, R. A. Raines, S. K. Rogers, "Malware Target Recognition," *United States Patent Application Publication [US 2012/0260342 A1]*, 2012.
- [15] Symantec Corporation, "Understanding Heuristics: Symantec's Bloodhound Technology," *Symantec White Paper Series*, vol. XXXIV, no. 1, pp. 1–14, 1997.
- [16] T. Dube, R. Raines, G. Peterson, K. Bauer, M. Grimaila, S. Rogers, "Malware target recognition via static heuristics," *Elsevier computers & security* 31 (2012) 137–147.
- [17] VX Heavens. (2010, Apr. 15). *Virus Collection* [Online]. Available: vx.netlux.org/v1.php
- [18] T. E. Dube, "A NOVEL MALWARE TARGET RECOGNITION ARCHITECTURE FOR ENHANCED CYBERSPACE SITUATION AWARENESS," *Air Force Institute of Technology, AFIT/DCE/ENG/11-07*, September 2011.