Malware Type Recognition and Cyber Situational Awareness

Thomas Dube*, Richard Raines*, Gilbert Peterson*, Kenneth Bauer*,
Michael Grimaila* and Steven Rogers†

*Air Force Institute of Technology, WPAFB, OH, USA
Email: {thomas.dube, richard.raines, gilbert.peterson
kenneth.bauer, michael.grimaila} {at} afit.edu

†Air Force Research Laboratory, WPAFB, OH, USA
Email: steven.rogers {at} wpafb.af.mil

Abstract

Current technologies for computer network and host defense do not provide suitable information to support strategic and tactical decision making processes. Although pattern-based malware detection is an active research area, the additional context of the type of malware can improve cyber situational awareness. This additional context is an indicator of threat capability thus allowing organizations to assess information losses and focus response actions appropriately. Malware Type Recognition (MaTR) is a research initiative extending detection technologies to provide the additional context of malware types using only static heuristics. Test results with MaTR demonstrate over a 99% accurate detection rate and 59% test accuracy in malware typing.

Index Terms

Invasive software, machine learning, security.

1. INTRODUCTION

Classic signature-based antivirus systems are effective at stopping global computer virus outbreaks, but are notoriously simple to avoid [1] for advanced local threats. Automated detection of such threats requires generic pattern recognition systems that determine program functionality using features common to dynamic or static analysis. Dynamic analysis is a behavior-based approach requiring runtime observation of samples in a protected environment. Static analysis is generally more efficient at detecting malware, because it uses high-level information, such as n-grams [2]–[8], strings [9] and metadata (data describing program structure and data) [9], [10].

Although pattern-based malware detection is extremely valuable, it provides no real threat context in the discovery of "malware" on a system with access to sensitive information. The additional context of the type of malware, such as *backdoor*, *keylogger* or *adware*, elevates the cyber situational awareness to an actionable level. The presence of a *backdoor* indicates that a competitor may have exfiltrated company secrets, while a *keylogger* may indicate a compromise of user credentials. Both of these discoveries require immediate, unique responses while the response to *adware* is likely trivial. Without the context of malware type, the information provided is not actionable.

Determining the type of new malware is difficult with current capabilities. Commercial products ship with low sensitivity heuristic scanners to avoid false positives. Customers can quickly lose confidence with highly sensitive scanners, not to mention they may expect the product to have the capability to "repair" any discoveries. None of the automated static analysis research [2]–[10] pursues the additional context of malware type. Additionally, dynamic analysis is orders of magnitude slower than static techniques and assumes the program reveals its true function within a short runtime duration.

Improving cyber situational awareness is the central objective of this research effort called Malware Type Recognition (MaTR, "matter"). Increasing awareness leads to improved mission assurance, because it provides leadership with the appropriate level of information for risk assessment and management. Recent static analysis research focuses exclusively on detection without investigating malware types. The resulting poor situational awareness forces leadership to make decisions without understanding associated risks.

MaTR extends [11] with additional malware types as well as detection to provide more threat context by using only static heuristics. MaTR examines the application of classic pattern recognition techniques used in static detection to determine types of malware while preserving high detection rates. MaTR achieves an apparent typing test accuracy rate of 59% across seven major malware classes, while maintaining a greater than 99% detection rate. These indicators can provide a significant improvement in cyber situational awareness over current methodology. The additional context can also help in the prioritization of analyst backlogs or more aggresive dynamic analysis techniques.

2. RELATED WORK

The most accurate static methods of automated malware detection come from n-gram research. Slicing a program into byte sequences of n bytes generates n-grams [2]. Non-malicious software and malware contain different frequencies of the byte sequences and the most unique become salient features for the classifier models.

IBM researchers Kephart and Arnold [2] are the pioneers in n-gram research as they define signatures automatically for new variants of known malware based on lowest likelihood to generate false positives for boot sector viruses. Tesauro et al [3] extend this work using neural networks with backpropagation and boosting with a voting system and incorporating cost function adjustments to reduce false positives. Arnold and Tesauro [4] describes the "cover" concept of finding n-grams present in all malware exemplars as both a feature and sample selection method. Szor states that the accuracy and false positive rate of this research were acceptable for Symantec to incorporate the IBM method into an early version of their commercial antivirus product [12].

Stolfo et al [7], [8] use n-grams to detect malware in document files and their implementation demonstrates superior detection performance over commercial antivirus scanners. Another distinction between this work and the IBM research is the use of low number byte character sequences (1- and 2-grams). Prediction uses k-means clustering with file type centroids, which the authors refer to as "fileprints". Experiments show that commercial antivirus software often fails to detect popular malware (even without additional obfuscation) when embedded in various places in document files.

Abou-Assaleh et al uses Common N-Gram analysis to classify malware [5]. The authors examine character n-grams found in executables in order to capture features associated with the author and tools used to write or compile the code. The authors make their observations on a small sample size of 25 and 40 samples of malicious and benign code respectively. The authors show that the character n-gram technique produced 100 percent accuracy on training data and 98 percent accuracy on test data.

Kolter and Maloof examine the results of several classifiers on malware detection via n-grams [6]. Techniques they test include naive Bayes, support vector machine, decision trees and boosted variants of each. In their experiments, they evaluate the classifier performance by computing the area under a receiver operating characteristic (ROC) curve. Their boosted decision tree model achieved the best accuracy, an area under the ROC curve of 0.996.

Using feature vectors of various information, including metadata, byte sequences and strings, as inputs, Schultz et al [9] test a variety of classifiers using 5-fold cross validation on unpacked malware samples. Their classifiers include inductive rule learning, multi-naive Bayes and naive Bayes classifiers. Although the multi-naive Bayes classifier with string inputs has the highest detection rate, the naive Bayes classifier with string inputs is within 1% with 99.43% and the best false positive rate of all tests (3.80%).

Treadwell and Zhou [10] use anomalies in the program header indicating the presence of packing as factors in a weighted risk assessment. They compute risk scores detecting malware that exceeds various threshold values. In the most sensitive tests, their model achieves a detection accuracy of 95.3% with a false positive rate of 3.872% on 144 malware samples.

All of these research efforts achieve high detection accuracies, but none examine malware typing. MaTR uses decision trees on hybrid static heuristic features to extend these techniques to malware typing while simultaneously demonstrating high detection rates. This additional context is a strong indicator of functionality, because the antivirus industry informally adopts naming and typing conventions from the Computer Antivirus Researcher's Organization (CARO) [13].

2.1. CARO Naming Standard

In order to establish naming standards and simplify sharing of information in the malware research community, CARO defines a recommended industry naming standard for discovered malware. The most recent naming standard specifies several malware types, including (in decreasing order of prioritization) *virus*, *dropper*, *Trojan*, *PWS* (password stealer, a.k.a. keylogger) and *backdoor* [13]. Although antivirus software vendors are not obligated to follow the CARO naming standard, it serves as a general guide for industry and researchers.

The above CARO type list demonstrates a potential problem with the naming standard and antivirus customer needs. The prioritization is researcher-centric, not customer-centric. Undoubtedly, the malware research community perspective of the problem is the containment of mass computer virus outbreaks similar to a biological virus and propagation has obvious criticality. The customer perspective, however, may be diametrically opposed. End users are generally more interested in the functional payload of a malware artifact rather than its replication intricacies—a local threat perspective vice a global one.

3. MODEL DESCRIPTION

MaTR uses a logical, two-phased approach by first identifying whether an executable sample is malicious and then determining its most likely type. Two forward, sequential feature selection searches identify the feature sets most significant to both detection and typing from a set of static heuristic features. The features in the set largely parallel those previously described in related research.

Malware detection is a straightforward two-class problem with distinct malware types grouped together under a singe malware class, *M*. The malware typing problem uses the distinct malware types provided by antivirus scans. The model dataset is a concatenation of static feature extractions from 32-bit non-malicious executables (*NM*) and malware of the following types: backdoors (*BD*), downloaders (*DW*), Trojans (*TJ*), password stealers (*PS*), worms (*W*), droppers (*DR*) and viruses (*V*). The sample corpus contains $40,498$ samples with $25,974$ malware samples from a recent update to [14]. The *NM* samples come from clean installations from vendor discs or digitally signed installation files.

The dataset examined is not multivariate normal. As a result, the classifier model selected for MaTR is the decision tree due to its flexibility when dealing with data from non-normal distributions and also categorical data. Many recent malware detection research efforts employ decision trees with great success [5], [6].

4. EXPERIMENT SETUP

The purposes of the following experiments are to verify classifier detection accuracy using MaTR's set of static heuristic features and to extend the application to malware typing. Both experiments are 3^2 factorial designs testing the effect of two significant factors in decision trees, minimum parent split and split criterion values. The three treatment levels chosen for the minimum parent split value are 10, 20 and 30 based on data from limited preliminary tests. The three treatment levels chosen for split criterion value correspond to the available functions in MATLAB: Gini's diversity index, the twoing rule and the maximum deviance reduction.

The minimum parent split value defines the cutoff threshold value for determining if the number of samples associated with a node in the tree warrants possible splitting. Lower cutoff values commonly lead to overfitting and loss of generalization [15]. On the other hand, larger minimum parent split values may prevent the tree from identifying significant patterns in the data or underfitting. The split criterion value maps to a MATLAB function the tree employs to measure impurity and determine the splitting feature and value.

Both experiments also use 5-fold cross validation with 50 replications. Immediately prior to each replication, a new random sampling determines the total set for subsequent cross validation runs. Preliminary experiment results indicate high consistency between both fold and replicate runs. To avoid overinflation of the detection results, sampling adheres to a fixed 2:1 *NM:M* ratio to accommodate decision tree model requirements. The malware samples are stratified random samplings from the malware types tested.

5. EXPERIMENT RESULTS

This section discusses the test results for the model parameter tests for both the detection and malware typing problems. Classifiers for the two problems demonstrate grossly different results with the detection and typing classifiers exceeding 0.99 and 0.59 apparent test accuracy rates respectively. Although typing accuracy is relatively low, it is impressive nonetheless as it provides threat context *without* the need for manual reverse engineering to malware detections and prioritization information for analyst backlogs. All ANOVAs and plots associated with these experiment results have 95% confidence intervals.

Table 1. Detection confusion matrix statistics.

Statistic	Producer Accuracy	Consumer Accuracy	Omission Error	Commission Error
NM	0.9949	0.9955	0.0051	0.0045
M	0.9909	0.9898	0.0091	0.0102

Table 2. ANOVA results for detection model.

Source	Sum Sq.	d.f.	Mean Sq.	F	p-value
A	3.02e-04	2	1.51e-04	88.4	1.09e-37
B	1.12e-05	2	5.61e-06	3.29	0.0376
A*B	8.30e-07	4	2.07e-07	0.121	0.975
Error	3.83e-03	2241	1.71e-06		
Total	4.14e-03	2249			

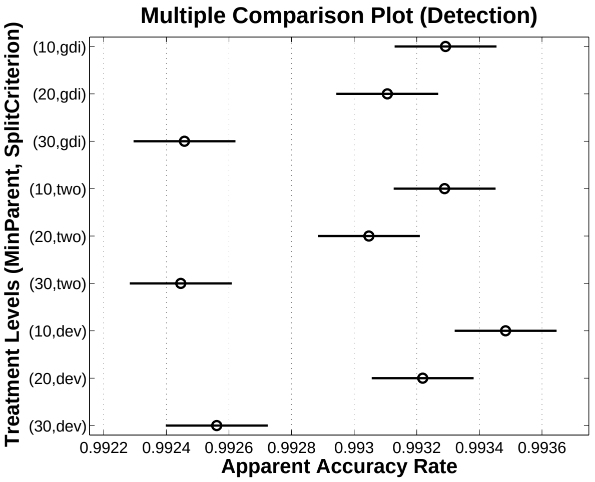

Figure 1. Comparison of detection parameters.

5.1. Detection Results

The results for the detection classifier are quite significant as all models exceed a 0.99 apparent test accuracy rate with equally impressive false positive and false negative rates. Table 1 shows the confusion matrix statistics for the trained model with the best treatment combination. The mean apparent test accuracy rate for this model over the cross validation and replication runs is 0.9935 with a kappa value of 0.9855, which demonstrates almost perfect agreement between model prediction and actual classes.

The commission error rate of 0.0102 for class *M* is low enough specificity to avoid deluging analysts with false positives. Furthermore, the commission error rate for class *NM* is low enough sensitivity to limit overlooking potential malware. Although this experiment uses a uniform cost of misclassification, a cost matrix adjustment can shift the false positive and false negative rates depending on operational needs.

The model parameter test examines the performance impact of two factors, minimum parent split value (A) and split criterion (B). According to the ANOVA results in Table 2, strong evidence leads to rejection of H_0 concerning equality of the main effects for factors A and B, but no evidence suggests a significant interaction effect between these factors. All ANOVA assumptions are met with residuals fitting a normal distribution and having constant variance.

Fig. 1 shows the mean comparison for different treatment level combinations with confidence intervals. Expectedly, the classifier performance improves as the minimum number of samples required for splitting decreases for the treatment levels tested. This trend is readily apparent in the mean comparison plot as the three lowest accuracies have the highest minimum parent split values (Factor A). All treatment levels with cutoff values of 30 have reduced performance that is statistically significant from all other treatment combinations.

The treatment level combinations with values for Factor A of 20 exhibit a decreased performance from points with values for Factor A of 10, but not enough statistical evidence exists to make a definitive claim of which is better. Both of these sets of combinations likely occur nearer to the "knee" of a performance curve that the minimum parent split values of 30, which may explain the significant difference between these data points and not the others.

The only remaining statistically significant treatment combination is $(10, dev)$, where the parameters are the values for factors A and B respectively. This treatment level combination yields the best result and is significantly different from the $(20, gdi)$ and $(20, two)$ combinations. This data point fits with another observable trend, the general improvement in performance for Factor B levels from *two* to *gdi* to *dev*.

These detection results show that MaTR demonstrates results similar to the other malware detection research using static analysis techniques and features. The next section describes the extension MaTR makes to the malware typing problem.

5.2. Typing Results

While seemingly not as spectacular as the detection tests, the malware typing model provides modest performance relative to the previous detection results. Considering the predictions do not require any lengthy manual inspection process and only use static heuristics, the typing results show a strong potential for

Table 3. Typing confusion matrix statistics.

Statistic	Producer Accuracy	Consumer Accuracy	Omission Error	Commission Error
BD	0.7510	0.7033	0.2490	0.2967
DW	0.6593	0.6366	0.3407	0.3634
TJ	0.3678	0.3589	0.6322	0.6411
PS	0.4553	0.4993	0.5447	0.5007
W	0.4283	0.4548	0.5717	0.5452
DR	0.3599	0.4282	0.6401	0.5718
V	0.5630	0.6742	0.4370	0.3258

prioritization of analysis backlogs. Table 3 shows the confusion matrix statistics for the model with the best treatment level combination. The mean apparent test accuracy rate for this model over the cross validation and replication runs is 0.5904 with a kappa value of 0.4738, which demonstrates moderate agreement between model prediction and actual classes. Although not extremely high, the expected value of a random classifier is only 0.1429 given seven malware classes tested.

Although the typing model fails to achieve high accuracy on the specific malware types, it still demonstrates potential value for identifying *BD*, *DW* and *V* classes. Classifier performance for identifying true *TJ* and *DR* samples is especially poor with producer accuracies of 0.3678 and 0.3599 respectively.

Antivirus applications often classify the same samples inconsistently, which can negatively affect the results of this typing experiment, because the supervised learning relies on the type labels from antivirus scans. Future investigation may examine the performance impact of confounding malware types together and vendor labeling disparities to regain high confidence in the results and maximize situational awareness. For instance, classifier inconsistencies between the *backdoor* and *Trojan* classes exhibit the highest error concentration and account for 5% of all typing errors.

Another explanation for the difficulty the model has classifying between malware types is a possibly inherent inadequacy in the salient value of the static heuristic feature set for making such determinations. The feature set may contain enough informational value to provide a high degree of accuracy for detection, but that may be its limit.

The model parameter test examines the performance impact of two factors, minimum parent split value (A) and split criterion (B). According to the ANOVA results in Table 4, strong evidence leads to rejection of H_0 concerning equality of the main effects for A and B, but no evidence suggests a significant interaction effect between these factors. All ANOVA assumptions for this test are met with similar residual normal and variance plots from the detection test.

Table 4. ANOVA results for typing model.

Source	Sum Sq.	d.f.	Mean Sq.	F	p-value
A	7.65e-02	2	3.83e-02	891	2.15e-285
B	2.49e-03	2	1.20e-03	28.9	3.89e-13
A*B	4.98e-05	4	1.24e-05	0.290	0.885
Error	9.62e-02	2241	4.29e-05		
Total	1.75e-01	2249			

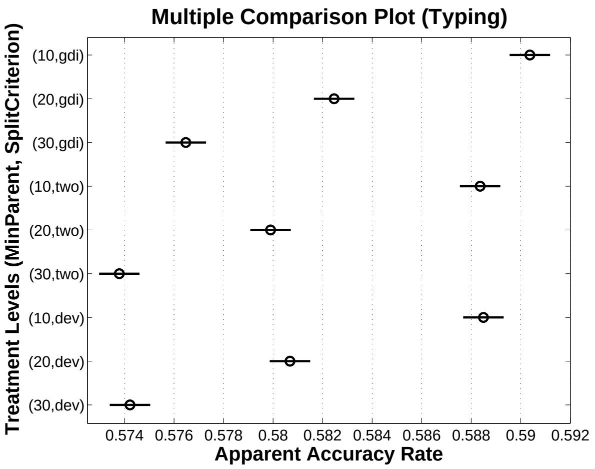

Figure 2. Comparison of typing parameters.

Fig. 2 shows the mean comparison for different treatment level combinations with confidence intervals. In this case, strong evidence suggests the treatment combination that exhibits the best performance mean for treatment combination $(10, gdi)$ is different. No significant difference exists between any of the two and dev levels for factor B with the same level for factor A (e.g., $(30, two)$ and $(30, dev)$).

This plot shows general patterns similar to the detection results. One observable trend is the general improvement in performance from levels two to dev to gdi for factor B, only slightly different than the detection results. Another general improvement trend is across treatment levels for factor A with lower split values exhibiting significant performance improvements.

5.3. Impact and Limitations

The high accuracy in generic malware detection provides a significant fine granularity capability advancement for baseline cyber situational awareness within local organization control. Adding a generic detection capability as an additional layer of defense adds additional capabilities. Additionally, typing information provides critical information to organizational leadership to consider available response options and future defense investments.

Using the classifiers in a two-stage sequence, the system detects most malware and additionally provides as much threat context as possible. In the hypothetical scenario of a malware on a system with access to sensitive company data, these experimental detection model most likely detects threat. The threat context provided by the experimental typing model has a high likelihood of identifying the backdoor allowing leadership to trigger the appropriate response action and observe threat activity. The same model has a moderate likelihood of identifying the keylogger.

Although this specific malware typing experiment did not yield a high confidence typing model against all types, a generic malware typing capability provides more detailed contextual information and retains significant value. Adjusting the cost functions can also improve technique specificity to target specific types of malware. All of this information clarifies situational awareness in regards to system and network integrity ultimately providing support for strategic and tactical level decision making. MaTR inherits all of the limitations associated with static heuristics.

6. CONCLUSIONS

Pattern recognition techniques can play a substantial role in malware detection and typing especially in cyber situational awareness and mission assurance. In exceedingly complex networks, simplifying assessment of operational readiness is a significant improvement and can lead to better risk management. MaTR demonstrates apparent malware detection accuracies above 99%. The performance results for the more difficult problem of malware typing are 59% overall, much higher than the expected value of 14% for a random classifier. The accuracies for identifying some malware classes are higher still.

Integration of this sensory data with other sensitive system and network sensors may further illuminate threat activity, and greatly improve upon defense-in-depth strategies. Avoiding all of these sensors simultaneously may prove too difficult for even the most determined adversaries.

References

[1] M. Christodorescu and S. Jha, "Testing malware detectors," in *Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis*. Boston, MA, USA: ACM, 2004, pp. 34–44.

[2] J. Kephart and B. Arnold, "Automatic Extraction of Computer Virus Signatures," in *Proceedings of the 4th Virus Bulletin International Conference*. Oxford, UK: Virus Bulletin, 1994, pp. 178–184.

[3] G. Tesauro, J. Kephart, and G. Sorkin, "Neural Networks for Computer Virus Recognition," *IEEE Expert*, vol. 11, no. 4, pp. 5–6, 1996.

[4] W. Arnold and G. Tesauro, "Automatically Generated Win32 Heuristic Virus Detection," *VIRUS*, vol. 51, 2000.

[5] T. Abou-Assaleh, N. Cercone, V. Keselj, and R. Sweidan, "N-gram-based Detection of New Malicious Code," in *Proceedings of the 28th Annual International Computer Software and Applications Conference*. New York, USA: IEEE, 2004, pp. 41–42.

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

[7] S. Stolfo, K. Wang, and W. Li, "Fileprint Analysis for Malware Detection," *ACM CCS WORM*, 2005.

[8] ——, "Towards Stealthy Malware Detection," *Malware Detection*, pp. 231–249, Mar. 2007.

[9] M. Schultz, E. Eskin, E. Zadok, and S. Stolfo, "Data Mining Methods for Detection of New Malicious Executables," in *IEEE Symposium on Security and Privacy*. IEEE, 2001, pp. 38–49.

[10] S. Treadwell and M. Zhou, "A Heuristic Approach for Detection of Obfuscated Malware," in *Proceedings of the 3rd International Conference on Intelligence and Security Informatics*. IEEE, 2009, pp. 291–299.

[11] T. Dube, R. Raines, G. Peterson, K. Bauer, and S. Rogers, "An Investigation of Malware Type Classification," in *Proceedings of the 5th International Conference on Information Warfare and Security*. Academic Publishing Limited, 2010, pp. 398–406.

[12] P. Szor, *The Art of Computer Virus Research and Defense*. Indianapolis, IN, USA: Addison-Wesley, 2005.

[13] V. Bontchev, "Current Status of the CARO Malware Naming Scheme," 2009, URL: www.people.frisk-software.com/ bontchev/papers/pdfs/caroname.pdf. Accessed Sept. 15, 2009.

[14] VX Heavens, "Virus Collection," 2010, URL: vx.netlux.org/vl.php. Accessed Mar. 28, 2010.

[15] R. Duda, P. Hart, and D. Stork, *Pattern Classification*, 2nd ed. New York, NY, USA: Wiley, 2001.