Malware Target Recognition of Unknown Threats

Thomas E. Dube, Member, IEEE, Richard A. Raines, Senior Member, IEEE, Michael R. Grimaila, Senior Member, IEEE, Kenneth W. Bauer, Member, IEEE, and Steven K. Rogers, Fellow, IEEE

Abstract—Organizations traditionally use signature-based commercial antivirus products as a frontline defense against malware, but advanced persistent threats craft custom malicious tools to achieve their objectives. Organizations safeguarding sensitive information have difficulty identifying new malware threats among millions of benign executables using only signature-based antivirus systems. This paper extends a performance-based malware target recognition architecture that currently uses only static heuristic features. Experimental results show this architectural component achieves an overall test accuracy of 98.5% against a malware set collected from operational environments, while three commercial antivirus products combine for a detection accuracy of only 60% with their most sensitive settings. Implementations of this architecture will enable organizations to self-discover new malware threats providing enhanced situation awareness for cyberspace operators in hostile threat environments.

Index Terms—malware detection, intrusion detection, antivirus, situation awareness, advanced persistent threat.

I. INTRODUCTION

Securing computer networks of large organizations is difficult, primarily due to their scale, scope and complexity. Analysts must examine a potentially overwhelming set of data to discover new malware threats. Attacker tools and malicious network traffic successfully hide in plain sight among millions of executable programs and billions of network connections, because organizations cannot effectively reduce these datasets to manageable levels.

Signature-based commercial antivirus and intrusion detection systems are effective for identifying known threats, but relatively ineffective against new unknown threats. To attackers, these systems serve as design constraints when designing new tools to avoid detection. Research demonstrates that commercial products are relatively easy to avoid [1]. Advanced Persistent Threats (APTs) as described by Bejtlich [2] certainly have this capability. Furthermore, these systems do not appreciably perform useful data reduction to reduce analyst workload as their detections are typically coarse binary “yes” or “no” outputs.

Malware detection research provides a potentially viable method of data reduction of analyst workload. Heuristic analysis techniques generally fall into two distinct categories: static and dynamic [3]. Static heuristics generally use non-runtime indicators [3], such as structural anomalies, program disassembly [4] and n-grams [5], [6], [7], [8], [9]. Alternatively, dynamic heuristics employ runtime indicators [3] normally obtained in virtual environments, such as commercial sandbox applications [10], [11], [12].

Despite the success that static heuristics enjoyed during the 1990s [3], today’s research heavily favors dynamic (or behavioral) heuristics [13], [14], [15], [16], [17]. Szor, a well-known industry expert, speaks to the current decline of static analysis techniques in his popular book [3]. Moser et al. presents limitations of static analysis techniques when generating a disassembly of program instructions [17]. Dynamic analysis methods suffer from limited operational utility due to slower runtime speeds than their static counterparts and incompleteness [18]. Their performance makes them operationally infeasible to test tens of thousands of unique programs on a single system, unless first using another method to prioritize workload. Dynamic heuristic analysis is also incomplete, because no guarantee of observing malicious activity exists.

While neither static or dynamic analysis alone is sufficient, the two can complement one another (and even commercial antivirus products) to provide a “full spectrum” defense against malware with reduced effective scan and detection time. Conceptually, sensitive and relatively fast static analysis methods can serve as prefilters for slower dynamic analysis methods reducing the effective runtime scan performance of the overall system while producing a lower number of false positives than either method alone. This research makes the following contributions:

1) Extends the Malware Target Recognition (MaTR) architecture initially proposed in [19] to an operational model for organization self-discovery of malware with low effective scan times and low false positive rates through successive data reduction and analysis.
2) Demonstrates generalization of the two static analysis models trained from [19] on a more current dataset of new “unknown” malware, and
3) Assesses the performance of three major commercial antivirus products against this same “unknown” malware set at different heuristic sensitivity levels.

These applications are the most challenging for research and commercial antivirus solutions as it represents new emerging threats that defensive systems did not consider prior to deployment. Furthermore, the test environment considers various levels of sensitivity simulating organizational execution in normal to hostile cyberspace environments. The MaTR static component prototype demonstrates the ability to find such malicious tools beyond the capabilities of another non-instruction based static heuristic (n-gram) model and the three commercial antivirus products tested at all sensitivity levels. Against a malware set collected from operational environments, the two static analysis methods far exceed the combined effective detection accuracies of the three major antivirus products tested.
The following sections describe related research, the MaTR architecture and its initial static component intended to serve as a highly sensitive pre-filter for future architecture components. The next section details the performance comparison of MaTR’s static component, an \( n \)-gram model and a suite of commercial antivirus products. Conclusions, a discussion of architecture limitations and a description of potential impact summarize this work.

II. RELATED WORK

The most recent static heuristic analysis research focuses primarily on the occurrence in programs of \( n \)-grams, which are byte sequences of length \( n \). Using \( n \)-grams as a feature source avoids the problems of generating pristine disassembly described by Moser [17], but these are certainly not the only non-instruction based static heuristics available. Researchers have also employed strings [3], [9], [20] as well as anomaly and structure data [3], [9], [21], [22] as features for malware detection classifiers. Currently, the scope of MaTR is strictly static heuristic analysis. This section briefly describes related research in static heuristic analysis of malware as well as detection evasion techniques.

A. N-gram Features

The IBM research of Kephart, Tesauro and Arnold provides the seminal research in \( n \)-gram analysis of malware [23], [24], [25]. 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 [23] for malware variants as well as for generic detection [24], [25].

While searching for methods to automate signature extraction for new variants of known malware, Kephart et al. discover the utility of \( n \)-grams for generic malware detection [23]. By determining the probability of finding specific \( n \)-grams in malicious and non-malicious programs, the authors fabricate a generic malware detection classifier.

Tesauro et al. successfully use neural networks to detect boot sector viruses [25]. They manipulate the decision threshold boundary to increase the cost associated with false positives as they cite that a single false positive reading likely affects thousands of systems. Despite significant computational and space constraints as well as a small sample size for training and testing, they achieve a false positive rate of less than 1% while detecting over 80% of unknown boot sector viruses.

They train the network with trigrams (3-byte length strings) that undergo a novel feature selection process. Initially, they canvas the entire sample corpus for trigrams and eliminate all that are common to both the malicious and non-malicious sets. Moreover, they reduce the list of trigram features to the set where each malicious training sample contains at least four trigrams. This selection process leads to a three order of magnitude feature reduction.

Expanding on their previous work, Arnold and Tesauro incorporate a voting system on multiple trained neural networks [24]. By training multiple networks with distinct features not used in others, they effectively avoid the major pitfall associated with heuristic scanners, high false positive rates. Their assumption is that these disparately trained networks rarely produce identical false positives. Szor cites that the Arnold and Tesauro network research has such a low false positive rate that Symantec incorporated it into its antivirus product default scanning [3].

Kolter and Maloof also made key contributions in \( n \)-gram research by examining the results of several machine learning classifiers on malware detection [7]. Techniques they test include naive Bayes, support vector machines, 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, a 95% confidence interval area under the ROC curve (AUC) of 0.9958 \( \pm \) 0.0024. The authors describe the difficulty of identifying why the presence of some byte strings combined with the absence of other byte strings contributes to high performance classifiers. In their study, they use 1,971 non-malicious executables from Windows 2000, XP operating systems, and SourceForge (http://sourceforge.net). Their malware sample set comprised of 1,651 samples obtained from the VX Heavens website (http://vx.netlux.org) and MITRE Corporation.

Among their findings, Kolter and Maloof identify best performance with an \( n \) = 4 byte sequence with a 1-byte sliding window. They treat the presence of an indicated \( n \)-gram as a Boolean feature to their boosted decision tree classifier. Tests utilize only the 500 most relevant \( n \)-grams based on information gain computations with the following formula:

\[
IG(j) = \sum_{v_j \in \{0,1\}} \sum_{C \in \{C_i\}} P(v_j, C_i) \log \frac{P(v_j, C_i)}{P(v_j)P(C_i)},
\]

where \( C_i \) is the \( i \)th class and \( v_j \) indicates the presence or absence of the \( j \)th \( n \)-gram. The prior and conditional probabilities are self-explanatory.

Abou-Assaleh et al. also use “character” \( n \)-grams, but with a smaller dataset of 25 malware and 40 non-malware samples [5]. They achieve training and test accuracies of 100% and 98% respectively.

B. Anomaly and Structural Features

Schultz et al. make key contributions by testing three different sources of features to identify malware [9]. In their first approach, they examine information from the portable executable (PE) header as features, such as import libraries and the number of imported functions from those libraries, with Cohen’s improved rule learning algorithm called RIPPER [26]. The second approach uses strings found in the binaries as features. Their third method captures byte sequences (presumably \( n \)-grams) expected to translate loosely to a representation of instructions, data, or both.

The authors used both the string and byte sequence data with naive and multi-naive Bayes classifiers. The results of the
naive Bayes classifier using string features is the most accurate classifier in their tests reaching a detection rate of 97.43% with a false positive rate of 3.80%. The authors concede that encryption (and presumably packing) obfuscates strings present in the executables [9], but the solution they suggest makes bold assumptions that can lead to high false positive rates. They propose that an effective method of handling the packing case is to initially assume that a sample is malicious and then if strings are found in the program the classifier defaults back to the naive Bayes algorithm.

Treadwell and Zhou use only structural anomalies as features to generate a weighted risk score [22]. They arbitrarily assign a risk score for each anomaly and use weights inversely proportional to the anomaly frequency in non-malware samples. In tests where the total risk score exceeds 1.0, they achieve a true positive rate (TPR) of 70.8% and a false positive rate of 3.872% on a sample set of 2,014 non-malware and 144 malware samples.

Rafiq and Mao extract hundreds of high-level program attributes and execute a feature selection process to determine the most salient features [21]. Using a naive-Bayes classifier with only the fifteen features selected, they exceed 90% detection accuracy with a 10% false positive rate. Like Treadwell and Zhou [22], their feature set does not include instruction level attributes, which avoids the problems with static heuristics identified by Moser [17].

C. Advanced Persistent Threat Motivation

Christodorescu and Jha [27] summarize the conflict between malware authors and researchers as an “obfuscation-deobfuscation” game, where advances from either side prompt retaliatory responses to at least maintain the status quo. Although an excellent observation about the tactics employed between two major players, it falls short of casting the strategic level problem for other stakeholders, specifically potential victims. As malware authors and researchers escalate in a cyberspace arms race, attackers continually exploit victims and conduct all manner of information operations against them. Antivirus systems are the most effective weapon to combat malware currently available for victims, but false expectations of their effectiveness abound.

Christodorescu and Jha [1] highlight difficulties commercial products have with handling simple obfuscation techniques, such as \texttt{nop} (a “no operation” assembly instruction) insertion and inserting unconditional branches (opaque predicates as described in [28]). In their test, they randomly apply these transformations to known malware samples that three commercial antivirus products initially detected. After transformation, they find that all three antivirus products failed to recognize ten unique mutations of the malware samples.

Advanced persistent threats enjoy nation state sponsorship [2] and can easily employ similar techniques to develop unique malware weapons that antivirus cannot detect. Major asymmetric advantages of the threat include unauthorized access to competitor sensitive data, low likelihood of discovery and prosecution, and low tool development cost. Ethics aside, the return on investment of this employment strategy is extremely high. The strategic level shortcomings of the status quo are the motivations for this current research.

III. Methodology

The MaTR architecture addresses the need to prioritize malware detection efforts from an organization perspective. The organization’s network can easily contain millions of unique programs, which is infeasible for a team of human analysts to examine. Furthermore, if dynamic analysis observes process behavior for a nominal 5 sec. period, it would take nearly two months to record the behavior of one million processes serially, not including analysis time of the generated data. The purpose of the MaTR architecture is to enable organizations to efficiently discover and respond appropriately to malware threats. It accepts program files as input, such as portable executable (PE) files common in the Microsoft Windows operating systems. The system implementation makes predictions that enhance the effectiveness of organization malware analysts.

The architecture is a performance-based hierarchy of the generic analysis process as Figure 1 depicts to reduce overall effective scan and detection times. This overall process is similar to airport security measures that use faster methods as prefilters before applying slower methods. Faster methods in airport security include prescreening of passenger names cross-referenced and manual checks of valid tickets before allowing passengers to proceed to the slower, but more definitive, methods. While these methods are slower, they generally provide higher confidence information for security personnel, such as scanning of carry-on luggage and the use of metal detectors. To let passengers pass directly to these slower methods makes the overall system less efficient and wasteful of limited resources. The same observation is true of employing dynamic and human analysis methods indiscriminately.

The general scan-time performance advantage of static heuristics makes them suitable for use as an initial filter for the potentially large set of all programs as shown in Figure 2. They can effectively reduce the set of all programs, $P$, to a more feasible candidate set $S$ for dynamic or human analysis. Dynamic analysis can further reduce the candidate set to subset $D$ to minimize false positives for the human analyst.
cell. Ideally, the relations \( n(P) \gg n(S) \geq n(D) = n(M) \) hold, where \( n(X) \) is the number of elements in the set \( X \).

![Fig. 2. Set of all programs with static and dynamic analysis and malware subsets.](image)

The ideal generic static heuristic classifier exhibits both high sensitivity and specificity. The high sensitivity leads to a lower number of false negatives, which means it is more likely to identify new threats. On the other hand, high specificity implies the method does not generate large numbers of false positives, which are pure (and measurable) wasted overhead for higher layers of the architecture.

For example, an organization has a total of one million unique portable executables on its enterprise network including 1,000 malware PEs. This organization uses only a dynamic analysis engine with false positive and false negative rates (FPR and FNR) of 0.75% each. The engine requires only 3 seconds to scan each sample. The expected scan time \((t_d)\), true positives \((TP_d)\) and false positives \((FP_d)\) using the dynamic analysis engine alone are calculated:

\[
t_d = \frac{(1E6 \text{ PEs})(3 \text{ s PE})}{86,400 \text{ s day}} = 34.7 \text{ days},
\]

\[
TP_d = (1,000 \text{ PEs})(1 - 0.0075) = 992.5 \text{ PEs}, \quad \text{and}
\]

\[
FP_d = (999,000 \text{ PEs})(0.0075) = 7,492.5 \text{ PEs}.
\]

If the same organization employs the proposed MaTR architecture and uses a static heuristic method that scans 2,500 PEs per second with a 1.5% FPR and FNR, the results change significantly. The scan time for the static component \((t_s(MaTR))\) is 400 s and its expected detections include 985 TPs and 14,985 FPs calculated as above. The dynamic analysis engine now scans only the resulting positives from the static component yielding (assuming independence):

\[
t_{d(MaTR)} = \frac{(985 + 14,985 \text{ PEs})(3 \text{ s PE})}{86,400 \text{ s day}} = 0.55 \text{ days},
\]

\[
TP_{MaTR} = (985 \text{ PEs})(1 - 0.0075) = 977.6 \text{ PEs}, \quad \text{and}
\]

\[
FP_{MaTR} = (14,985 \text{ PEs})(0.0075) = 112.4 \text{ PEs}.
\]

A few important observations are in order at this point. The effective scan time \((t_s(MaTR) + t_d(MaTR))\) using the MaTR architecture is a 98.4% improvement over using a dynamic analysis engine alone. Also, the overall expected TPR with MaTR drops only 1.5%, but the MaTR architecture generates 98.5% less false positives in this scenario. This scenario is realistic as malware dynamic analysis tends to have higher accuracy than static analysis.

Components at the lower levels of the hierarchy complement those at higher levels in other ways as well. For instance, a single static analysis component may target typical malware defenses \([3, 21, 22]\) in order to reduce the dataset effectively for a higher level component or another static analysis component. Additional static components may target other indicators of malware that do not employ typical defenses. In this manner, multiple alert paths from input to presentation to human analysts can exist within the architecture as shown in Figure 3. In this figure, circles represents each component with directed edges indicating the direction of the alert flow. This flexibility provides organizations with an agile system for adapting to new threat tactics.

![Fig. 3. MaTR alert paths change for different threat environments.](image)

In different threat environments, such as those identified by various levels of the Department of Defense’s Information Operations Conditions (INFOCON) \([29]\), new alert paths can become activated. Components along existing paths can also alter their internal sensitivities when the INFOCON level changes. Section IV explains how these experiments test the MaTR static analysis component against an n-gram classifier and three commercial antivirus products at various sensitivity levels.

### A. Static Analysis Component

Dube et al. evaluated the detection performance of the first static analysis component of MaTR against sets of 25K non-malicious and 31K malware samples \([19]\). This static component targets typical malware defenses that result in program structural anomalies when compared to non-malware. The component followed a standard machine learning process for malware detection using only non-instruction based static heuristics. While many researchers and commercial companies use this same source of data, none rely exclusively on it and achieve the performance levels of this MaTR component \([3, 7, 9, 21, 22]\).

1) **Bagged Decision Tree Classifiers:** This static analysis component employs bagged decision trees as its classifier based on its performance on pilot studies, which parallels findings of other researchers using similar data sources \([7, 8]\). The decision tree is a machine learning classifier with a tree data structure. Classification decisions are the result of
traversing from the tree root to a leaf node. Each non-leaf node employs a split variable and split value to determine the path of traversal to a leaf node. Traversal to a leaf node results in a final class assignment based on prior probabilities from the remaining sample subpopulation at that leaf established during training.

The MaTR component tested uses tree data resulting from training with the MATLAB bagged decision tree implementation TreeBagger [30]. Training decision trees involves determining the proper feature and value for each node to split the training set into subpopulations with lower node impurity (a measure of the subpopulation consisting of same class labels). For each decision split, TreeBagger by default randomly selects $\sqrt{n}$ features of $n$ total features as candidates for the split variable and assesses them using one of the following impurity functions: Gini’s diversity index, the twoing rule and the maximum deviance reduction [30].

Bagging, or bootstrap aggregation, is an augmentation method of the performance of a single tree by generating a tree ensemble with each tree based on different bootstrap samplings. During training, the selection of $n$ samples from the training set (of size $n$) with replacement constitutes a bootstrap sampling. The resulting classifier uses a majority vote of the individual trees in the ensemble.

The MaTR component used in these tests uses an ensemble of 25 trees with default parameters. By default, TreeBagger considers $\sqrt{n}$ random features for each cut variable, allows a minimum of one observation per leaf node and employs the Gini’s diversity index as an impurity measure [30]. Other defaults include no pruning and using equal misclassification costs [30].

2) Feature Set: A key distinction between this MaTR component and other commercial and research products is its feature set. This component achieves high detection performance despite restricting its features exclusively to non-instruction based static heuristics. Instead of following a mathematical model to determine features, it utilizes anomaly and structural heuristic features commonly used by analysts [3], [21], [22] when examining samples to determine if they are indeed malicious. Rafiq and Mao found that malware routinely contains structural anomalies (78%), while non-malware does not (5%) [21].

Currently, this MaTR component utilizes over 100 static heuristic features based on structural anomalies and structural information itself. Many of the component’s features are interval or rational giving each feature more expressive power, unlike other methods which use exclusively Boolean features [7], [8]. The n-gram model uses information gain calculations to compute features, which results in a set of “weak” features as opposed to “strong” features identified by Rafiq and Mao [21]. The expressiveness of interval and rational features as well as the value of “strong” features explains the improved performance of this component over the n-gram model.

This component does not attempt to generate an instruction disassembly due to the difficulty of validating its correctness [17] nor does it use instruction sequence signatures as commercial antivirus programs commonly use [18]. As the MaTR component does not rely on complex computations of large samplings to determine the final feature set, it avoids the overhead of a resource-intensive feature selection step such as Kolter and Maloof found [7], [8].

Structural anomalies are generally logical operations on program header information or its references. Classes of structural anomalies include: section names [3], [21], [22], section characteristics [3], [21], [22], entry point [3], [22], imports [9], [21], [22], exports [22], and alignment [3]. Structure information, included to enable classifiers to identify additional anomalous combinations, comes directly from headers. A description of the more useful anomaly features for this MaTR static analysis component follows.

a) Non-standard section names: Several researchers [3], [21], [22] identify the presence of non-standard section names as anomalous. Microsoft [31] defines several standard section names for PEs and many compilers adopt this standard for default compiling. This standardization has led to an overwhelming majority of non-malware containing only standard section names. According to Rafiq and Mao [21], only 3% of non-malware use unconventional section names, while 80% of malware samples use non-standard names.

b) Non-standard section characteristics: Many researchers [3], [21], [22] identify non-standard section characteristics as an anomaly. If a code section has read, execute and write characteristics instead of the normal read and execute characteristics, it immediately raises analysts’ suspicions. Normally, the program uses sections with these permissions to unpack obfuscated code before attempting to execute it. This particular anomaly is common in malware, because packing is a common malware armoring technique [3]. Other examples are sections with atypical characteristics, such execute characteristics for the .rsrc section.

c) Entry points: A program entry point that points to a section not marked as containing code is anomalous [22]. Szor states that program entry point anomalies include not pointing to a code section (.text for default compiling) or pointing to the last section [3]. Packers commonly adjust the entry point to point to an additional code section to start the unpacking process.

d) Imports: Inclusion of information regarding import libraries and functions is common among malware research [3], [9], [21], [22]. Common features include the numbers of import libraries and functions. Executables with a low number of imported functions are suspicious [22], because programmers normally provide program utility by importing functions to perform I/O, encryption or complex math operations. Malware samples often exhibit approximately the same number of imports as the introductory “Hello World” program, which is suspicious given the malware’s larger file size.

e) Exports: Treadwell and Zhou also identify dynamically-linked libraries that export no functions as anomalous [22]. Since the purpose of a dynamically-linked library is to provide functionality to other programs via exported functions, the absence of exported functions is surely suspicious.
IV. Experiment Setup

The purpose of these experiments is to show performance comparisons between the malware detection methods at various levels of sensitivity against a set of relatively unknown malware samples. The malware detection methods tested include the trained MaTR static analysis component and trained n-gram model from [19] and three commercial antivirus products. Tests on this particular set of malware have strategic significance for organizations, as these samples are representative of tools that an advanced persistent threat may employ.

A. Data Collection

This experiment uses trained models from previous work [19] against a set of relatively unknown malware obtained from private sources. The following subsections describe the datasets used in the previous work [19] as well as the unknown malware dataset for these tests.

1) Initial MaTR Static Component Datasets: The initial MaTR component and n-gram models trained in [19] used 32-bit PE samples of non-malware and malware obtained from well known sources. All “clean” samples came from harvesting of clean installs of Microsoft Windows XP, Vista, and Windows 7, while the malware samples came from an updated download of the VX Heavens dataset [32]. Specifically, the malware samples were Trojan, worm, and virus types as identified by the antivirus label assigned to them. Training and testing used 25,195 clean and 31,147 malware samples (representing approximately 5K malware families) from these sources. Table I shows the number of samples used for training and testing from each major malware category and subcategory. Table II lists several malware families represented in the original dataset. These malware categories, subcategories and families are extracts of information from the malware labeling of Kaspersky Anti-virus used to index [32]. The specific MaTR static analysis component and n-gram models chosen for this validation test on the “unknown” malware set were the highest accuracy performers for each detection method from a series of 10-fold cross validation runs from [19].

2) Unknown Malware Dataset: The “unknown” malware set for these tests is a combined collection of 278 new malware threats discovered by incident response teams from anonymous organizations. These entities shared this data with the stipulation that their identities remain hidden. With this agreement, duplication of these experiments with the same exact dataset is not possible, but other validation tests against other new malware samplings should yield similar results. Section V includes justification for the extrapolation of these results to other new malware sets.

B. Experimental Design

The following experiments are an application of the MaTR static component and n-gram models trained from previous work [19]. Both methods used identical training and test sets during initial development [19]. Table III shows the trained model results on previous test data with a decision threshold of 0.50. In particular, the receiver operator characteristic (ROC) area-under-curve (AUC) exceeds the resulting n-gram models from the original Kolter and Maloof work (AUC 0.9954 ± 0.0024) [7] and a retest of their method with a larger and more current dataset (AUC 0.999173 ± 0.000248) [19]. The accuracy and FPR and FNR also are favorable to other research results [9], [25]. Previous MaTR work elaborates on these findings [19]. The static heuristic model from [19] could serve as a near ideal first-pass filter in the MaTR architecture considering its high sensitivity and high specificity given the previous discussion of ideal characteristics. Research of additional static and dynamic analysis components as well as human analysis methods is future work.

This experiment has two factors: the detection method and the sensitivity threshold for the detection method. The detection method has five levels, the trained MaTR component and n-gram models from [19] and each of the three commercial antivirus products. Table IV lists the three commercial antivirus products tested, but subsequent results show only
randomized labels for each antivirus product to avoid unofficial endorsement. The sensitivity factor has three distinct levels, which are specific to each detection method. For the antivirus products, the factor levels are the built-in low, medium, and high heuristic scanner sensitivity levels. For the MaTR and n-gram models, varying the decision threshold of the boosted decision tree classifier to 0.75, 0.50, and 0.25 for the positive class constitutes a simple “standard” for low, medium, and high sensitivity levels.

These tests are side-by-side comparisons of the MaTR and n-gram models trained in [19] and three commercial antivirus products against the malware validation set. (The identities of the three commercial products is masked to avoid the appearance of an official endorsement of a specific product.) For consistency with prior research, these tests both adopt a standard experimental design using stratified, ten-fold cross validation. Each disjoint fold contains roughly the same number of malware samples from the validation set. During each run, a different fold functions as the test set. Additionally, the 10-fold cross validation is replicated ten times with different folds for each replication. All models and the commercial antivirus products use identical folds across both factors to reduce experimental variance. Furthermore, the MaTR and n-gram models under test did not use any samples from the “unknown” malware set during their training or testing [19].

All antivirus products use default configurations, except for the sensitivity factor levels, and have updated signatures as of the date of the experiment. Each detection method runs in its own Windows 7 virtual machine with identical baseline configurations. The unknown malware samples are on a CD-ROM that each scanner examines in turn.

C. Measures of Effectiveness

Since the following experiments only evaluate detection results against a malware set, the most appropriate measure of effectiveness is the true positive rate (TPR). The accuracy for this test is synonymous with TPR, because the test does not consider any negative class samples. Computed confidence intervals result from studentized bootstrapping with the sampling defined by the cross-validation folds. Dube et al. includes false positive rate results for these static analysis models [19] as this measure is outside the scope of these experiments.

V. RESULTS AND DISCUSSION

Although this experiment has two factors, a full analysis of variance is not necessary as the results are readily apparent. Therefore, discussion of results follows the sensitivity factor levels, including descriptions of results for all detection methods for each.

<table>
<thead>
<tr>
<th>Product</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symantec Endpoint Protection</td>
<td>11.0.5002.333</td>
</tr>
<tr>
<td>Avira AntiVir Premium</td>
<td>10.0.0.663</td>
</tr>
<tr>
<td>Kaspersky Anti-virus 2010</td>
<td>9.0.0.736</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Detection Method</th>
<th>Mean TPR</th>
<th>95% Confidence Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaTR</td>
<td>0.938783</td>
<td>0.930447 – 0.947119</td>
</tr>
<tr>
<td>n-gram</td>
<td>0.870504</td>
<td>0.858084 – 0.882922</td>
</tr>
<tr>
<td>AV1</td>
<td>0.456825</td>
<td>0.439162 – 0.474489</td>
</tr>
<tr>
<td>AV2</td>
<td>0.388439</td>
<td>0.370665 – 0.406214</td>
</tr>
<tr>
<td>AV3</td>
<td>0.356071</td>
<td>0.340914 – 0.371229</td>
</tr>
</tbody>
</table>

Table V shows the mean scan times for each detection method (except the n-gram model as no full working prototype was available) on the entire 278 samples in the unknown malware validation set. All detection methods execute on identical virtual machines, except for software installation, with similar loads and scan malware samples on a CD-ROM. Although not intended as a definitive performance evaluation, the scan times demonstrate a general execution time performance advantage for MaTR. The runtime disparity is likely due to the fact that MaTR encapsulates a generic signature for malware and does not attempt to identify the specific strain of malware, a traditional requirement for antivirus products. In the table, the abbreviation AV refers to a commercial antivirus and the following number is a randomly assigned index consistent across sensitivity tests.

A. Low Sensitivity Level

For test runs with the low sensitivity level, the MaTR component’s performance is superior to all other methods tested by a substantial margin as shown in Table VI. While the best commercial antivirus product fails to reach a 50% TPR against the unknown malware validation set, the MaTR component correctly identifies nearly 94% and the n-gram model predicts 87% of the positives correctly. The low sensitivity level of this test and the discrepancy of results highlight the significant advantage of generic malware detection over current commercial solutions. Furthermore, this advantage occurs when restricting classification decisions to non-instruction based static heuristic features.

The MaTR component and the n-gram model also demonstrate the lowest variance across the ten replications of 10-fold cross validation as evidenced by their narrower confidence intervals. MaTR exhibits the lowest variance of all methods tested. The confidence intervals for the antivirus products are nearly twice the width of MaTR’s intervals. While the commercial products exhibit less consistency across folds and replications than MaTR, they are consistent with each other. MaTR’s TPR is more than double the best commercial product’s rate.

Examination of the MaTR static analysis component feature importance confirms those discussed in Section III-A2. Several
features are at most rarely used by the component model, but the prototype retains these features for future assessments of tactical changes. Examples of these features are the non-standard section characteristics unobserved in the initial training set for specific sections. Change in adversary tactics to use the currently unobserved sections in a non-standard manner is trivial to achieve. Removing these features from the model achieves no substantial goal.

B. Medium Sensitivity Level

The results from the medium sensitivity level are surprisingly similar to those at low sensitivity, especially for the commercial products. Only the MaTR and the n-gram models along with AV1 exhibit any change in detections at all (see Table VII). At this level, the MaTR component finds 13 more samples in the entire validation set boosting its TPR to over 98%. Meanwhile, the n-gram model finds 22 more samples, while AV1 detects 3 additional samples as malware overall increasing its TPR to nearly 47%. The remaining antivirus products exhibit no change in detections at the medium sensitivity level.

Interestingly, the MaTR model becomes even more consistent across the experiment folds and replications as evidenced by a 50% reduction in the width of its confidence interval. The n-gram model is also more consistent reducing its confidence interval width by 33%. Although AV1 also increases its detection rate at the medium sensitivity level, it continues to exhibit roughly the same consistency across folds and replications. Again, the other antivirus products exhibit no change in detection rates.

In previous work [19], the MaTR and n-gram models both performed well against known malware samples at the default “medium” sensitivity. These tests demonstrate the robustness of these two methods against unknown malware. Based on the performance results from previous work [19] as shown in Table III, the decrease in TPR for the MaTR and n-gram models are only 1.37% and 3.72% respectively. Although both are far more robust against unknown malware than commercial products, the MaTR component exhibits minimally degraded performance.

C. High Sensitivity Level

Tests at the high sensitivity level produce the starkest contrasts between the generic signature methods and the commercial antivirus products as shown in Table VIII. The MaTR static analysis component misses only 1 of the overall 278 samples in the validation set and achieves a TPR of over 99%. The n-gram model misses a mean of 5 samples increasing its TPR to over 98%. Remember, the training and testing of the MaTR and n-gram models did not include any of the malware samples from this malware validation set, which adds substantial strategic value to this finding. Again, the MaTR model demonstrates more consistent performance across folds and replications as it reduces the resulting confidence interval width by an additional 50%.

AV1 detects only 1 additional sample from its performance at the medium sensitivity level. Although AV1 is the best performing commercial antivirus product tested, it fails to reach a TPR of 50% against this set of unknown malware. The remaining antivirus products tested at their highest sensitivity levels exhibit no change in performance from their lowest sensitivity levels against this unknown malware set.

Although a surprising result, several theories may explain why the commercial antivirus products are relatively sensitivity invariant to this set of unknown malware samples. Malware authors may have more thoroughly tested detection from these commercial products due to their market share or targeting purposes. The antivirus companies have also made design decisions regarding trade-offs between false positives and sensitivity. Regardless of the explanation, the test results are unanticipated.

As the final planned test, the detection rates of MaTR and the n-gram model are more than two times greater than the best performing commercial antivirus product tested. Against an operational validation set, this finding is significant as it demonstrates the capability gap of commercial products versus unknown malware. These threats can maneuver around commercial capabilities and rely on organizations’ often blind trust in antivirus products to protect them as they have little other visibility into malware discovery. Increasing defensive maneuver may stop or degrade these threat activities.

D. Union of Antivirus Products

Due to the large disparity in results for the various sensitivity levels, additional tests examine the combined effectiveness of the three commercial antivirus products. In this case, the union of the three commercial products forms a “super” antivirus detection system. If any of the component products detect a sample, then the test considers the conglomerate system to have detected the sample.

Subject to the same experimental design, the union of all detections from commercial antivirus against the validation set still does not achieve similar performance levels as the MaTR or n-gram models. Table IX shows the comparative results of this component and the union of antivirus products at all sensitivity levels. The union performs substantially better.
than the individual products with a 10 to 25% TPR boost in detection performance. The same performance inconsistencies across folds and replications as seen in individual antivirus product tests is visible in the union results as well. In spite of its improvements versus the individual products at all sensitivity levels, the union still fails to appreciably challenge the performance of the MaTR or n-gram models.

The union of antivirus products exhibit no change from medium to high sensitivity levels. In this case, the additional sample detected by AV1 in the earlier high sensitivity level test, at least one of the remaining products already found. The collective improvement from low to medium sensitivity levels for the union of antivirus products is lower than the individual improvement of AV1 due to the same rationale.

Figure 4 is a Venn diagram showing the overlap of detections from the three commercial antivirus products for all sensitivity levels. The Venn diagrams indicate a 25% overlap of all three products and at least partial overlap of 37% of the entire sampling. Given the substantial overlap of detections from the commercial products, the most likely conclusion is that a substantial portion of these samples were merely new malware samples when initially discovered by organization incident response teams. Table X shows the Kaspersky-labeled malware families (as was Table II) detected from all antivirus products in these tests. The emphasized entries in the table indicate newer samples of the same malware families included in the original training dataset in [19]. This observation also implies that these experiments are repeatable on new malware datasets with similar methods.

At this point, one may conjecture that commercial antivirus products, while relatively effective at containing global or regional threats, are not nearly as effective against targeted, local threats. Put another way, traditional antivirus is helpful for containing global outbreaks of viruses and worms, but is not as valuable for protecting an organization from targeted malware, such as an advanced persistent threat may use. These threats target specific victims with custom malware that avoids detection by antivirus products and is generally not available for antivirus analysis.

From the antivirus industry perspective, they cannot protect against malware they have never seen. Users expect a commercial antivirus product to not only alert the user of an infection, but also to sanitize or clean the malware. If threats use malware customized to avoid detection from current antivirus systems and they restrict distribution, only the victim organization has substantial opportunity to find the offending sample. Even if discovered, the victim may not desire to share the sample with malware researchers, because its an indirect acknowledgment of system or network compromise.

Given the apparent value of generic malware detection methods against new malware tools, organizations should consider investing in an in-house capability. This approach may be most prudent for organizations with sensitive data or in extremely competitive markets.

VI. Conclusions

The commercial antivirus products tested are heuristic sensitivity invariant to unknown malware samples with one product detecting only 4 additional samples and the others exhibiting no change for different levels. Determined APT threats will alter their tactics to accomplish their missions when presented with defensive obstacles, such as static antivirus signatures. Given the highly maneuverable nature of competitive threats, flexibility in detection is paramount and the antivirus products tested fail dramatically in this application. Commercial antivirus scanners lack the sensitivity to provide organizations with elevated cyberspace situation awareness given the dynamic complexity of modern networks. Without this knowledge, organizations cannot accurately determine the reliability of their information and communications technologies to successfully accomplish their missions.

The MaTR static component’s detection rate on the set of unknown malware exceeds 98% with a medium sensitivity level, whereas the best commercial antivirus product tested achieves less than 47%. With high sensitivity levels, the MaTR model’s detection rate exceeds 99%, while the best commercial product barely reaches 47%. The union of antivirus products with high sensitivity fails to reach a 60% detection rate against this strategic dataset. Another static heuristic method, n-grams, outperforms the commercial antivirus products, but does not achieve the MaTR component performance levels.

Although antivirus products still play a vital part in an organization’s defense in depth, machine learning techniques can play a substantial role in malware detection especially related to cyberspace situation awareness and mission assurance.

### Table IX

<table>
<thead>
<tr>
<th>Detection Method</th>
<th>Mean TPR</th>
<th>95% Confidence Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaTR (low)</td>
<td>0.938783</td>
<td>0.934047 – 0.947119</td>
</tr>
<tr>
<td>Union of AVs (low)</td>
<td>0.589894</td>
<td>0.571996 – 0.607792</td>
</tr>
<tr>
<td>MaTR (med)</td>
<td>0.985569</td>
<td>0.981508 – 0.989829</td>
</tr>
<tr>
<td>Union of AVs (med)</td>
<td>0.597090</td>
<td>0.579471 – 0.614709</td>
</tr>
<tr>
<td>MaTR (high)</td>
<td>0.996402</td>
<td>0.994249 – 0.998555</td>
</tr>
<tr>
<td>Union of AVs (high)</td>
<td>0.597090</td>
<td>0.579471 – 0.614709</td>
</tr>
</tbody>
</table>

### Table X

<table>
<thead>
<tr>
<th>Malware Families</th>
<th>Small</th>
<th>Agent</th>
<th>VB</th>
</tr>
</thead>
<tbody>
<tr>
<td>NetSky</td>
<td>Qhost</td>
<td>VBNA</td>
<td></td>
</tr>
<tr>
<td>Inject</td>
<td>Krap</td>
<td>Zbot</td>
<td></td>
</tr>
<tr>
<td>LdPinch</td>
<td>Jorik</td>
<td>Genome</td>
<td></td>
</tr>
<tr>
<td>Invader</td>
<td>SkSocket</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In exceedingly complex networks, simplifying assessment of operational readiness is a significant improvement and leads to better risk management. The MaTR component’s high TPR implies that it accurately detects malware and does not often mistake it for benign software, including unknown malware. Given its high TPR and low FPR from [19], this MaTR component shows operational data reduction utility to provide real strategic value for decision makers by providing situation awareness indicators of threat activity on information and communications technologies.

A. Limitations

In an operational environment, the FPR of this MaTR static analysis component likely will increase. The MaTR architecture can accommodate this with subsequent automated static and dynamic analysis methods before presenting the human malware analyst with its predictions. While lower level components of the MaTR architecture should be sensitive enough to avoid false negatives, subsequent reductions of the dataset should focus on minimizing potential false positives.

A deployed system implementing the MaTR architecture may be susceptible to reverse engineering and exploitation, but this is an oversimplification of the attacker’s situation. The MaTR static component tested targets common malware defenses. These same defenses are the bane of antivirus companies as it protects malware against detection by their products. Focusing MaTR components in this manner can create a symbiotic relationship with commercial antivirus products, where avoiding detection by MaTR or a commercial antivirus product creates a vulnerability to detection by the other.

Also, an analysis of a MaTR system assumes a decentralized deployment. A centralized employment of a MaTR system makes it less vulnerable to adversary exfiltration and analysis. This approach forces attackers to evade an organization-specific adaptable defensive system blindly, unlike avoiding commercial antivirus products.

B. Impact

The high accuracy in generic malware detection provides a significant fine granularity capability advancement for cyberspace situation awareness within complete local organization control. Given the TPRs of the MaTR static analysis component, n-gram models and current commercial antivirus products, a static heuristic malware detection method is a potentially “game changing” technology for identifying most likely threats among millions of unique executables. By effectively reducing analyst overhead, systems that implement the MaTR architecture can literally eradicate the asymmetric advantages on which attackers rely and level the cyberspace battlefield. It also provides critical information to enable organizational leadership to consider available response options and future defense investments.

C. Future Work

Future work will examine performance characteristics of dynamic analysis methods to verify effective improvement in malware detection performance and analyze human-machine interface detection 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 will also expand testing of other static heuristics classifiers as prefilters for the MaTR architecture or to provide specific context to satisfy operator needs.

References


Thomas E. Dube is a major in the United States Air Force and an Assistant Professor of Computer Engineering in the Department of Electrical and Computer Engineering at the US Air Force Institute of Technology. His research interests include network and system integrity, malware analysis, operating systems, and reverse engineering. Dube has an MS in information assurance and PhD in computer engineering both from the Air Force Institute of Technology. He is a member of Eta Kappa Nu, Tau Beta Pi, and IEEE.

Richard A. Raines is a Professor of Electrical Engineering in the Department of Electrical and Computer Engineering at the US Air Force Institute of Technology. His research interests include computer communication networks, global communication systems, intrusion detection systems, and software protection. Raines has a PhD in electrical engineering from Virginia Polytechnic Institute and State University. He is a member of Eta Kappa Nu and a senior member of the IEEE.

Michael R. Grimaila is an Associate Professor of Systems Engineering at the Air Force Institute of Technology (AFIT). He received his PhD (1999), MSEE (1995), and BSEE (1993) all from Texas A&M University. He holds the Certified Information Security Manager (CISM) and the Certified Information Systems Security Professional (CISSP) certifications. Dr. Grimaila’s research interests include computer and network security, cryptography, mission assurance, and systems engineering. He is a member of the Association for Computing Machinery (ACM), Information Systems Audit and Control Association (ISACA), International Information Systems Security Certification Consortium (ISC2), a Senior Member of the Institute of Electrical & Electronics Engineers (IEEE), and is a Fellow of the Information Systems Security Association (ISSA).

Kenneth W. Bauer is a Professor of Operations Research at the Air Force Institute of Technology where he teaches classes in applied statistics and pattern recognition. His research interests lie in the areas of automatic target recognition and multivariate statistics. Dr. Bauer holds the following degrees: BS, Miami University (Ohio), 1976; MEA, University of Utah, 1980, MS, Air Force Institute of Technology, 1981; PhD, Purdue University, 1987.

Steven K. “Capt Amerika” Rogers is a Senior Scientist at the Air Force Research Laboratory where he serves as the principle scientific authority for Automatic Target Recognition and Sensor Fusion. Dr. Rogers’ research focuses on qualia exploitation of sensor technology, QUEST. Dr. Rogers founded a company for developing practical applications of advanced information processing techniques for medical products. The company invented the world’s most accurate computer aided detection system for breast cancer. He has over 150 technical publications and 20 patents. Dr. Rogers is a Fellow of the IEEE and SPIE.