Scalable Telemetry Classification for Automated Malware Detection

Jack W. Stokes[1], John C. Platt[1], Helen J. Wang[1], Joe Faulhaber[2], Jonathan Keller[2], Mady Marinescu[2], Anil Thomas[2], and Marius Gheorghescu[2]

[1] Microsoft Research, Redmond WA 98052, USA,
{jstokes,jplatt,helenw}@microsoft.com,
[2] Microsoft Corp., Redmond WA 98052, USA
{joefa,jkeller,mady,anilth,mariusg}@microsoft.com

Abstract. Industry reports and blogs have estimated the amount of malware based on *known* malicious files. This paper extends this analysis to the amount of *unknown* malware. The study is based on 26.7 million files referenced in telemetry reports from 50 million computers running commercial anti-malware (AM) products. To estimate the undetected malware, a classifier predicts the underlying nature of unknown files recorded in the telemetry reports. The telemetry classifier predicts that 69.6% (4.27 million) of the unknown files are malicious. Assuming the unknown files predicted to be malicious by the classifier are malware, the telemetry classifier also allows us to estimate the efficacy of the AM system indicating that signatures detected 82.8% (20.6 million) of the malicious files. We have validated our system by conducting a longitudinal study to measure the false positive and false negative rates over a period of thirteen months.

Keywords: malware classification, telemetry, sample collection, prefilter

1 Introduction

The anti-malware (AM) industry faces two significant problems in the battle to protect their customers' computers from being infected by malware. First these companies are paradoxically confronted by the challenge of trying to discover malware in huge amounts of telemetry data while only having samples (i.e. copies) of a small fraction of the unknown files hosted on their users' computers. Typically, commercial AM products transmit telemetry reports from a large percentage of their client computers when users try to download or install known or potential malware. Although our company receives over 100,000 new file *samples* every day which need to be analyzed, we had previously collected samples from only 3.1% of the files referenced in the set of telemetry reports received in October 2010. Ideally, AM companies would have a copy of every unknown executable file observed on their clients' machines. In this scenario, analysts clearly cannot investigate each new file manually, and anti-malware companies

must automate detection of these threats. Researchers have described systems showing great promise on automatically detecting malware [4], [6], [32]. However, in-depth analysis can be time consuming. In some cases, Anubis [4] can require several minutes to analyze each file [2]. Each month, we receive telemetry reports corresponding to tens of millions of files that were not detected as being malicious by the AM system's signature detector. Even with distributed processing and only analyzing new, incoming files, clearly these in-depth malware detection systems will struggle to analyze all of the new, potentially malicious files. Second the companies cannot accurately measure the performance of their systems in detecting the amount of malware in the ecosystem. Consumers often see claims that an anti-malware product detects X% of malware, and industrial reports provide a glimpse of the amount of malware detected on user's machines [13], [24]. However these estimates are based on detections of *known* malware. What is missing is the true detection rate, including the *unknown* files, found in the wild. While it is impossible to measure this malware detection rate exactly, we seek a better estimate of the amount of malware in this study.

To address these problems, we propose a lightweight telemetry report classifier shown in Figure 1. AM clients transmit telemetry reports to the backend system running the telemetry classifier. Each report contains metadata, including file and machine identifiers, associated with the installation or scan of an untrusted portable executable (PE) file including application binaries, screen savers, drivers, and Active X controls. Unlike previous work, we assume that the majority of the executable files cannot be accessed directly since they are either located on remote computers or the installations were blocked by the signature detector. To help manage the collection and analysis of unknown files, the telemetry classifier assigns a probability that the file associated with the report is malicious solving two problems. The reports can be ranked to determine which files are more likely to be malicious for collection from the remote users' computers. In addition, the telemetry classifier output allows the system to prefilter the queue of unknown files for timely processing by an in-depth malware analysis system. Files with the highest malware probability are collected or analyzed first. Recently, several systems have been proposed for prefiltering (i.e. ranking) samples for in-depth analysis [18], [26], [34], but these systems require a sample of the file for classification and clustering on the results of static analysis. In our system, the AM client can perform both static and *dynamic* analysis of the file on the remote computer, and we use this telemetry information for prefiltering. Although we only utilize a simple behavior feature in this study, more sophisticated dynamic execution features could also be detected and reported by the client. Furthermore, the performance of the signature detectors can be measured by the telemetry classifier. Unlike previous work, our estimates include *unknown* files which have not been previously detected. Using the proposed system, we estimate the percentage of malicious files encountered in a large sample of 26.7 million telemetry reports, each corresponding to a unique file, received from a population of over 50 million computers.

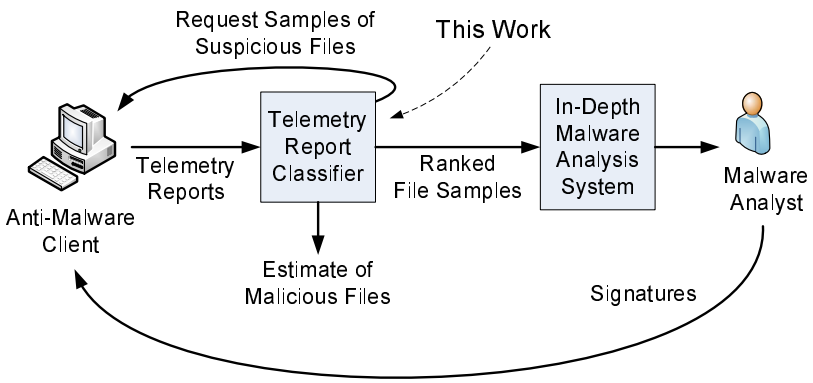

Fig. 1. Classifying malware on the backend based on metadata reports generated by anti-malware clients.

A key aspect of this work is to investigate whether *individual* telemetry reports can be accurately classified to predict if an unknown file residing on a remote client is malicious or benign. Previous work [22], [30] proposes various types of malware classifiers, but these systems assume access to the PE file. Recently, commercial products including Symantec's AM products [7], [8] and Microsoft's Internet Explorer have started using reputation data to detect malware or benign files and consider files which have been frequently observed, but not detected as malicious, to be benign. The approach followed in this paper does not rely on file prevalence information. This choice was made explicitly to identify polymorphic and zero-day attacks. In addition, the earlier file-based studies were conducted on small training sets; while these preliminary efforts were promising, it was unclear if the results scale with large numbers of labeled data. In this study, we use over 253 thousand, labeled telemetry reports to train and test the telemetry classifier.

To handle the large number of reports used for training, the system utilizes several technically significant components. The telemetry classifier uses a *limited* combination of features (Section 2) derived from the telemetry data including static features from the binary file and one, simple behavioral feature indicating what action caused the report to be generated. Among these, using tri-grams of the file's locality sensitive hash is a novel implementation which scales well. Excluding this feature from our model decreases the accuracy by over 18% (Section 4). We use a feature selection algorithm based on 2x2 contingency tables and the mutual information criterion to create the datasets (Section 3). Next we describe several algorithms used to train both linear and non-linear classifiers for analyzing the telemetry reports and highlight several machine learning algorithms for the security community (Section 4). Logistic regression trained with the L-BFGS algorithm and including L1 and L2 regularization performs best for our task. A boosted decision tree algorithm trained with the MART criteria also

Feature	Description
File Name	Name of the PE file
Original File Name	File name in the original report
File Name Matches Original?	Does the file name in this report match the file name in the original report?
File Type	What type of file is it?
Signer Name	What organization signed the file?
Signing Authority	What certificate authority issued the signature?
Signature Type	Was the file signed or not? If signed, is the signature legitimate or invalid?
Description	What is the description of the file in the header?
Organization	Manufacturer of the binary file
Version	Version number of the binary file
LS Hash	Locality sensitive hash
Behavior Feature	Represents the simple behavior that caused the report to be generated

Table 1. Summary of the telemetry report attributes and additional data aggregated by the backend system.

performs well. Lastly, an approximation to the SVM using L-BFGS optimization is competitive and can require much less time to train compared to exact methods (e.g. sequential minimal optimization). We validate the telemetry classifier in a thirteen month longitudinal study to measure the false positive and false negative rates of the samples received one month after training.

We implemented the lightweight telemetry classifier and used it to estimate the number of malicious executable files (Section 5). The telemetry classifier predicts that 69.6% (4.27 million) of the unknown files involved in the reports are malicious. Although biased on the computers which sent the telemetry, this estimate gives a better sense of the total amount of malware. The telemetry classifier also allows us to estimate the efficacy of the signature detector. Assuming all of the unknown files predicted to be malicious by the classifier are indeed malware, the telemetry classifier indicates that signatures detected 82.8% (20.6 million) of the malicious files. A summary of the contributions of this paper includes:

- A large-scale system to classify anti-malware telemetry reports is proposed and implemented, and the results are presented. Using tri-grams of locality sensitive hashes is a novel feature for the system.
- The number of malicious executable files and the effectiveness of a suite of anti-malware products are estimated from a sample of 26.7 million telemetry reports received from over 50 million computers.
- We demonstrate that the lightweight telemetry classification system can be used to prioritize files for sample collection and prefilter these samples for more in-depth analysis.
- Training classifiers based on six different algorithms including logistic regression with L-BFGS optimization and L1 and L2 regularization as well as an approximation of the linear SVM are highlighted for the security community.

2 Telemetry Metadata and Features

This study is based on a large collection of telemetry reports received from a suite of commercial anti-malware products. Our company manufactures these

security products which detect and remove spyware (e.g. Windows Defender) as well as viruses and other malware (e.g. Forefront Client Security, Windows Live OneCare, Windows Live Security Scanner, and the Microsoft Malicious Software Removal Tool). Our analysts utilize the reports in our efforts to detect new malware on personal computers (PCs) running the Windows operating system.

The telemetry reports consist of various attributes measured by the AM client running on the remote computer when it detects that a new file is being installed or a previously undetected file is running. To limit the number of reports received at the backend web service, only telemetry reports corresponding to files which have not been signed by a *trusted* certificate authority are transmitted. Additional information can be constructed at the backend by examining the telemetry reports across all of the reporting clients. This local and backend metadata corresponding to the (potential) installation of a file is summarized in Table 1. All of the attributes are extracted by the AM client with the exception of the second and third rows (i.e. original file name and file name matches) which are determined on the backend. In addition to the low-level features discussed in this section, we also construct other features indicating if a particular attribute is blank or null. For example, if the organization is null, a boolean feature is set to true. This high-level metadata is discussed in more detail below and serves as the basis for the low-level features used to train the classifiers in Section 4.

In addition to the metadata listed in Table 1, the telemetry reports also contain several unique file identifiers including the SHA1 and MD5 files hashes. While these hashes cannot serve as features for any classification system since a small change in the executable file leads to a large change in the corresponding hash value, they allow us to assign a label to the incoming telemetry report for files which have previously been collected, investigated and categorized (e.g. malware, benign) by analysts. We could potentially also use files detected by anti-virus signatures and include these in our dataset but we have not done so for the following reason. Training primarily with samples determined by the signature detector may lead to a situation where the telemetry classifier learns to only recognize files we currently detect; doing so may prevent the telemetry classifier from identifying files not currently identified by the signature detector.

While Table 1 illustrates the high-level metadata found in the telemetry reports, we cannot directly use this information as the features for the telemetry classifier described in later sections. There are too many values associated with some of the attributes in the table. For example, we measured over 71 million distinct file names in one month of telemetry data. Next, we describe the methods used to transform this metadata into a set of potential low-level features for the telemetry classifier we train in Section 4. This transformation is just the first step in determining the final classifier features. We further restrict (i.e. filter) this set of low-level features through feature selection in the following section. Only two features may vary when comparing telemetry reports from a unique malware sample, namely, the file name and the behavior which caused the report to be generated (described later). In addition to the file name associated with the report, the telemetry classifier also considers other derived features. All string

Name	Percentage
Not Signed	95.68%
Freeze.com, LLC	0.26%
Zango	0.21%
WebDevAZ, Inc.	0.16%
WHENU.COM INC	0.15%

Table 2. Most frequent malware signer names.

Name	Percentage
Not Signed	78.53%
Microsoft Windows Component Publisher	5.07%
Microsoft Windows	3.54%
Microsoft Corporation	2.50%
Microsoft Windows Publisher	2.50%

Table 3. Most frequent benign signer names.

features are efficiently encoded using 1.5 grams. Given the shear number of unique strings in the data, we cannot represent each string as a feature. To limit the total number of possible features, we propose a compromise we call 1.5-grams (pronounced one and a half grams). To compute a 1.5-gram set for a unicode string, we first convert the string to a byte array. The initial 1.5-gram for the array is determined as the first 12 consecutive bits (i.e. 3 nibbles). To compute the second 1.5-gram, we slide the index by 4 bits, and the 1.5-gram is the value of next 12 consecutive bits. For standard 8-bit text, this encoding represents a full character and half of the following character. It can also fully represent more obscure non, 8-bit characters. Using the 1.5-gram representation only requires 4096 (2^{12}) possible values for all possible file names. A separate feature identifies if the file name in the report matches the file name associated with the original report of the executable. Furthermore, the type of file (e.g. keyboard driver, printer driver, application, DLL) is also used as a feature.

Two important features of the system are which organization signed the file and which certificate authority granted the certificate. These features were also suggested by Nachenberg *et al.* [7]. In addition, the certificate is verified for authenticity. The signature type feature indicates whether or not the file was signed. If it was signed, was the signature valid? Tables 2 and 3 provide an example of the most frequent organizations that signed the files associated with malicious and benign reports, respectively; these tables do not necessarily describe the most discriminant (i.e. best) features for the telemetry classifier. For example, since 95% of the malware and 78% of the benign files are not signed, a signature value of "Not Signed" will not be a good feature. While it is not surprising that most malware is not signed, the results for benign files is an artifact of the telemetry reporting process. Reports are not sent by the AM clients for files which are signed by trusted organizations. Thus, the signed, benign files in the head of the distribution are not reflected in this data. The Microsoft signatures in Table 3 are most likely particular signatures used on a small number of files, and therefore, these signatures have not been added to the list of trusted certificates.

Another important feature is the certificate authority (CA) which granted the certificate, and the data for the CA is listed in Tables 4 and 5. Interestingly, a small fraction of malware authors have managed to obtain certificates granted from respectable CAs. The reason is that they are trying to provide assurance to

Name	Percentage
No Issuer	95.66%
VeriSign Class 3 Code Signing 2004 CA	1.16%
Thawte Code Signing CA	1.15%
UTN-USERFirst-Object	0.59%
INVALID:Thawte Code Signing CA	0.52%

Table 4. Most frequent malware certificate authorities.

Name	Percentage
No Issuer	78.49%
Microsoft Windows Verification Intermediate PCA	7.88%
VeriSign Class 3 Code Signing 2004 CA	4.51%
Microsoft Windows Verification PCA	3.55%
Microsoft Code Signing PCA	2.24%

Table 5. Most frequent benign certificate authorities.

the users that the code is legitimate. This behavior indicating attackers trying to build trust has been studied recently for website certificates [10]. We encoded each distinct value for the signer and certificate authorities in the set of potential features.

All PE files contain information in the header such as the manufacturer, description, and version number. This data is transmitted to the backend in the telemetry reports and encoded as features using 1.5-grams for the telemetry classifier. In addition to the SHA1 hash, a locality sensitive hash (LS hash) is also computed for the file by the AM client and transmitted to the backend. Unlike standard hashes which completely change when a single bit in the file is altered, LS hashes have the property that changing a small amount of code introduces a small change in the resulting hash. Bayer, *et al.* [3] and Jang, *et al.* [19] have utilized the LS hash as a feature for malware clustering. Clustering, however, requires comparing pairs of LS hash values which can be computationally expensive. In our design, the telemetry classifier uses tri-grams of each file's LS hash which is a novel feature representation. The LS hash tri-grams from variants of malware families in the training set increase the likelihood that these tri-grams are associated with malicious files. As a result, training and evaluation are not adversely affected as the scale increases. It should be noted that since the LS hash is composed of hexadecimal digits, only 2^{12} features are required to represent all possible tri-gram values.

The action that caused the report to be generated is also used as a feature for the telemetry classifier. There are roughly 50 distinct behavior actions found in our telemetry reports, and these behavioral actions are indicative of a file being installed on the computer. Table 6 provides the five most frequent behaviors associated with a report generated by malicious files while the five most frequent behaviors associated with benign files are given in Table 7. Example behavior actions include installing an ActiveX control, Browser Helper Object, or driver, adding a Run Key to automatically start a program each time the user logs on, starting a process, or scheduling a task. For both malware and benign files, downloading an ActiveX control is the main behavioral feature associated with the telemetry reports. Surprisingly, only 2.93% of malicious reports were associated with Browser Helper Objects.

Name	Percentage
ActiveX Downloads	73.93%
Run Keys	9.56%
Running Processes	3.77%
Browser Helper Object	2.93%
Task Scheduler	1.96%

Table 6. Most frequent malware behavioral features.

Name	Percentage
ActiveX Downloads	70.64%
Services	6.26%
Drivers	5.02%
Run Once Keys	3.91%
Run Keys	3.48%

Table 7. Most frequent benign behavioral features.

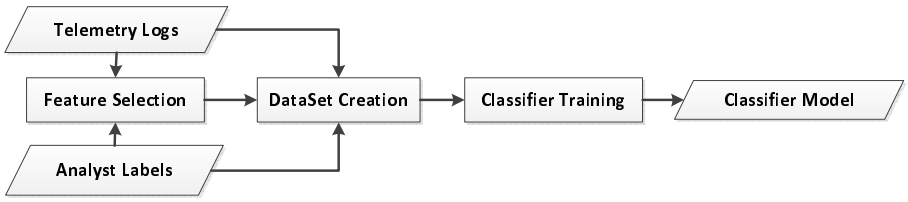

Fig. 2. Anti-Malware telemetry classification training system.

3 Feature Selection and Dataset Creation

In this section, we describe the process used to create the dataset required to train the telemetry classifier. The anti-malware telemetry classification training system is illustrated in Figure 2. The raw telemetry logs and analyst labels described earlier are input to the system which includes three processing blocks: feature selection, dataset creation, and classifier training. An integral step in the process is feature selection which excludes potential features that are not beneficial during classifier training. In the previous section, we transformed the high-level, telemetry metadata into a large number of potential low-level features, but we cannot use all of these to train the telemetry classifier. The feature selection algorithm determines the most discriminant (i.e. best) subset of all of the features to be used for classification. Based on the selected features, a labeled dataset is next constructed from the analysts' labels and the low-level encoded features derived from the telemetry reports. In the next section, we then use the labeled dataset to train the telemetry classifier using several different algorithms. The output of the training system is a classifier model (i.e. a set of weights or parameters) which can be used to predict if unknown reports were transmitted due to malware or benign files.

Constructing a dataset from all of the encoded data from the previous section can lead to hundreds of thousands of potential features. Using too many low-level features can cause overfitting which is due to training a complex machine learning algorithm with an insufficient number of training examples. If the model is too complex, the results when the system is deployed to production may be significantly worse compared to those observed when trained and tested on a small labeled dataset. A general rule is to select the number of features F

for the system to be the total number of samples divided by a sufficiently large number (e.g. 8-10). The feature selection algorithm we use first computes a 2x2 contingency table for each potential feature based on the mutual information criterion [23]. A maximum likelihood estimate of the mutual information criterion serves as our ranking score $R(f)$:

$$R(f) = \frac{D}{N}\log_2\frac{N\cdot D}{(\hat{BD})(\hat{CD})} + \frac{B}{N}\log_2\frac{N\cdot B}{(\hat{AB})(\hat{BD})} + \frac{C}{N}\log_2\frac{N\cdot C}{(\hat{CD})(\hat{AC})} + \frac{A}{N}\log_2\frac{N\cdot A}{(\hat{AB})(\hat{AC})}$$

where A is the number of times the potential is not in the reports and the file is determined to be benign, while D is the number of malicious reports which include the potential feature. B (C) similarly is the report count for malicious (benign) files not including (including) the potential feature. In addition, $\hat{AB} = (A + B)$, $\hat{AC} = (A + C)$, $\hat{BD} = (B + D)$, $\hat{CD} = (C + D)$, and $N = A + B + C + D$. Finally, the top F features are selected from the highest ranked mutual information scores.

4 Telemetry Classifier Performance

Now that we have created our labeled dataset in the previous section, we turn to the task of training our telemetry classifier. In this section, we investigate the performance of five, *linear* and one, *nonlinear* classification algorithms. We are particularly interested in linear classifiers because they are fast to train, but more importantly, they can be used to evaluate unknown reports very quickly. Since tens of millions of reports are received every day, evaluation of each unknown report must be fast.

We first consider two forms of logistic regression [5] trained using stochastic gradient descent (LR-SGD) and L-BFGS (LR-L-BFGS) [1] as the optimization methods to learn the model parameters. Next, we train a support vector machine (SVM) [5] with a linear kernel based on the Pegasos [31] algorithm (SVM-Pegasos) as well as an *approximation* of the linear SVM [36] again using L-BFGS (SVM-L-BFGS). The final linear classifier considered in this study is the averaged perceptron [11]. We also train with a nonlinear algorithm employing boosted decision trees using the MART [12] algorithm. Boosting has previously been suggested for malware classification [22], [28].

To train and test the classifiers, we created a labeled dataset from 253,517 telemetry reports consisting of 173,548 malicious reports and 79,969 benign reports collected over a four month period ending January 2012. We selected a single telemetry report to represent each distinct file, as represented by a unique SHA1 hash. To evaluate the performance of the six classification algorithms, we use 5-fold cross validation which is the most fair way to do so. In cross validation, the entire labeled dataset is split equally into N (e.g. 5) sections. For each fold, we use one section as the test data and combine the remaining sections as

the training data. After the telemetry classifier has been trained and evaluated for all folds, every sample in the dataset has been independently used in the testing set. To be completely fair, we also rerun the feature selection algorithm for each fold's training data. Consequently, we never do feature selection using samples from the test set. For these experiments, the number of selected features was determined as one-tenth of the number of samples used for training. Using 4/5 of the total 253 thousand samples leads to the selection of 20,281 low-level features.

Since we are dealing with a binary (i.e. two-class, malware versus benign) classification problem, we investigate the performance of the classifiers using detection error trade-off (DET) curves which plot the false negative rate versus the false positive rate for the 5-fold cross validation results. Figure 3 shows the DET curves for the six different classification algorithms. In addition, the equal error rates, where the false positive and false negative rates match, and the training time for one fold of the cross-validation are shown in Table 8. Of the six different algorithms, LR-L-BFGS outperforms the remaining classifiers, particularly at lower false positive rates. This version of logistic regression includes both L1 and L2 regularization terms. For L1 regularization, the algorithm tries to force small weights to have a value equal to zero which helps improve the algorithm's ability to generalize to new telemetry reports. In this case, the L1 and L2 parameters are each set to 1.0. MART is competitive and is slightly better than LR-L-LBGS at higher FP rates. SVMs have been well studied in the machine learning literature [5], [15]. However, training an SVM for large data sets can take a prohibitive amount of time. Zhang, *et al.* [36] proposed an approximation to the linear SVM based on a modified version of logistic regression. The central idea is that the SVM's non-linear hinge loss can be approximated by the logistic regression's smooth log-loss function. We often use this algorithm to approximate the SVM in our work since the datasets tend to be very large. This SVM approximation trained with L-BFGS (SVM-L-BFGS) also performs reasonably well compared to LR-L-LBGS and MART. However, this implementation does not include a separate L1 regularization term which may contribute to the decrease in detection accuracy compared to LR-L-BFGS. The averaged perceptron and LR-SGD are competitive, but the SVM trained using the Pegasos algorithm performed significantly worse compared to the other five algorithms. We conducted another experiment to verify the contribution on the proposed LS Hash features on the LR-L-LBGS model. Removing the LS Hash features from the model increased the CV equal error rate for one particular dataset from 5.76% to 6.81%, an increase of over 18.2%. As Table 8 shows, the classifiers are reasonably fast to train on a large server with dual, 2.0 GHz Intel E7540 processors and 128 GBs of RAM. The best performing algorithm, LR-L-BGGS, requires approximately 11 minutes to train. This training time is approximately one-fourth of the time required to train the second best algorithm, MART, which is significantly more complex.

In Figure 4, we analyze how the telemetry report training set size affects the equal error rate for three of the algorithms: LR-L-LBGS, MART, and LR-SGD.

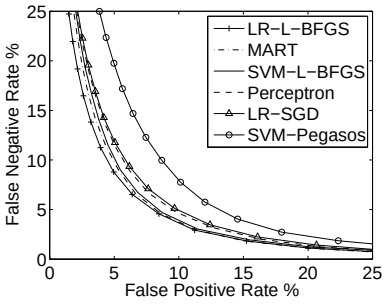

Fig. 3. DET curves for the malware classifiers trained with several different algorithms.

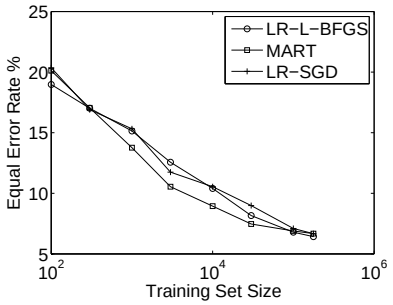

Fig. 4. Equal error rates for classifiers trained with various training set sizes.

Algorithm	Equal Error Rate (%)	Training Time
LR-SGD	7.41	00:07:52.53
LR-L-BFGS	6.45	00:11:13.21
SVM-L-BFGS	6.77	01:59:32.89
SVM-Pegasos	10.34	00:01:12.59
Averaged Perceptron	7.21	00:02:54.15
MART	6.64	00:44:58.39

Table 8. Equal error rates and training times for six different telemetry classifier algorithms.

The motivation for studying the effect of the training set size is that the results presented in [22] and [30] are based on small training set sizes of 3622 and 4301, respectively. We would like to understand if a particular classification algorithm is the main factor in determining the classification performance or if the amount of training data is more important. The figure clearly shows that increasing the training set size leads to a significant decrease in the equal error rate for all three models. For many of the different sample sizes, MART performs best but is surpassed by LR-L-BFGS starting at 100 thousand samples. The relative performance of LR-L-LBFGS and LR-SGD depends on the sample size. As the sample size increases, the equal error rates become very close for the different algorithms. One important result from Figure 4 is that the test error is still decreasing even with a training set of 157 thousand samples. Although since the x-axis is on a log scale, achieving better accuracies requires higher and higher numbers of training samples.

Next in Table 9, we evaluated the performance of the telemetry classifier over a period of thirteen months on new, unique reports (i.e. files) received in the month following the classifier training but were not included in the training set. For each test month, the telemetry classifier was trained on the previous five months of unique labeled reports. The files associated with both the training

Start Training Month	Final Training Month	Test Month	CV Equal Error Rate (%)	Test FP Rate	Test FN Rate
2010_09	2011_01	2011_02	7.66	9.73	9.45
2010_10	2011_02	2011_03	7.52	9.87	8.94
2010_11	2011_03	2011_04	7.43	13.36	6.63
2010_12	2011_04	2011_05	7.37	11.46	8.14
2011_01	2011_05	2011_06	7.43	18.14	16.17
2011_02	2011_06	2011_07	7.29	8.0	10.33
2011_03	2011_07	2011_08	7.23	7.44	11.07
2011_04	2011_08	2011_09	7.40	8.97	10.56
2011_05	2011_09	2011_10	7.13	10.77	14.61
2011_06	2011_10	2011_11	6.37	8.87	7.56
2011_07	2011_11	2011_12	6.33	6.95	9.12
2011_08	2011_12	2012_01	6.17	21.11	6.48
2011_09	2012_01	2012_02	6.78	8.35	6.11

Table 9. Longitudenal results for the telemetry classifier system over a period of 13 months.

and test reports were previously determined to be malicious or benign by either manual analysis by professional analysts or other automated means. For example, computing the SHA1 hashes of all files in an off-the-shelf program and adding them to a whitelist constitutes an automated method of determining a file's label. As noted earlier, we did not rely on detections based on AV signatures for this validation. The table shows that cross validation rates on the training sets are fairly consistent. However, due to a smaller amount of data, the FP and FN rates for the test month have a larger variance and are almost always higher than the cross-validation error on the training set. This is to be expected because it is the most difficult test on previously unseen data. In January 2012, the FP rate was 21.1% which can happen due to a small number of samples labeled benign for a particular test month. In this case, the number of true positives was 29,836 and true negatives was 5,931. In addition, we found 1,587 false positives and 2,068 false negatives. Overall, there is reasonable agreement between the test and training CV errors. For collecting new samples and sample submission to an in-depth processing system, we believe these error rates are acceptable. In these scenarios, a false positive results in the collection or submission of roughly one benign sample for every nine malicious samples.

5 Malware Estimation

Now that we have trained our malware telemetry classifier to predict the label of the unknown reports, we can use the classification system trained with LR-L-LBFGS to estimate the number of malicious files. The 26.7 million telemetry reports in our sample can be divided into several high-level sets as shown in Figure 5. The figure indicates the number of reports received in October 2010

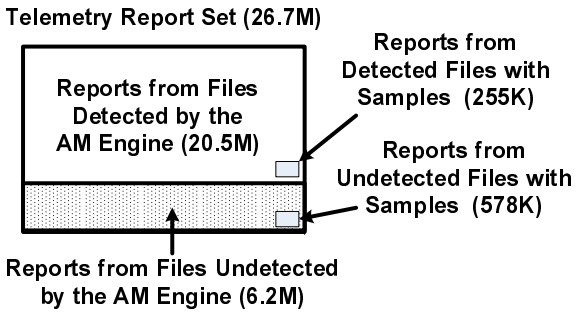

Fig. 5. Anti-malware telemetry report description. All reports represent a distinct executable determined by a unique SHA1 hash.

considering only a single report for each distinct executable file. In other words, we sample the most recent report received for each file and disregard each previous report containing that particular SHA1 file hash. These reports are divided into two sets, namely those for files identified by the signature detector running on the remote computer (20.5 million) and those for unknown files which were not detected (6.2 million). One of the main goals of this paper is to predict the label of these undetected reports in the shaded box. Furthermore, detected reports include 255 thousand reports corresponding to executables where a sample of the file has been previously collected at the backend. Similarly, the file collection system includes 578 thousand file samples found in the undetected reports. From the figure, we see that we have only collected samples of files found in 3.1% ((255K + 578K)/26.7M) of the reports thus providing motivation to analyze reports in the absence of a file sample.

To estimate what percentage of unknown files are predicted to be malicious, we first estimate the percentage of malicious executables from the unknown reports in our sample population. The second and third columns of Table 10 provide the number of reports based on the three detection methods, manual labeling, signature detection, and telemetry classifier prediction. In total, we received reports for 26,749,556 distinct, files during October 2010. Of these 0.27% (72,771) were generated due to files where we have a sample which has been previously labeled as malicious by an analyst. These files are part of the set of 255 thousand file samples in the detected reports in Figure 5. Similarly, we observed that 0.07% (19,952) of the reports correspond to files labeled as benign by analysts from the 578 thousand reports for which we have samples. The vast majority (20,518,412, 76.7%) of reports were due to files that were detected as malicious by the engine but not labeled by the analysts.

For the remaining reports for undetected files, we now use the telemetry classifier to predict how many of these were caused by malware and how many correspond to benign files. Similar to the system described in Section 4, we trained a telemetry classifier using the labeled samples up through October 2010 and then

used it to predict the label of the unknown reports. The equal error rate for this version of the telemetry classifier is 5.9%. The third row of Table 10 shows that 16.0% (4,270,521) of the unknown reports are predicted to be malicious, while 7.0% (1,867,900) are predicted to involve benign files. *The telemetry classifier predicts that 69.6% (4,270,521/(4,270,521+1,867,900)) of the reports from undetected files are malicious.* Furthermore, *93% of the total reports correspond to malware.* The telemetry classifier output also allows us to estimate the efficacy of the signature detector. The table indicates that *signatures detected 82.8% (20,591,183/24,861,704) of the malicious files observed on the client computers.*

Detection Type	Malicious	Benign
Manual Labeling	72,771 (0.27%)	19,952 (0.07%)
Signature Detection	20,518,412 (76.7%)	Not Applicable
Telemetry Classifier Prediction	**4,270,521** (16.0%)	**1,867,900** (7.00%)
Total	24,861,704 (93.0%)	1,887,852 (7.1%)

Table 10. Statistics for distinct files associated with telemetry reports received in October 2010.

6 Discussions

We believe the results in Section 4 are quite encouraging. After conducting the experiment to measure how well the telemetry classifier can predict if the file associated with the telemetry report is malicious or benign and reviewing the results, we next set up a research web service for analysts to classify telemetry reports from client machines. The system accomplishes several tasks, namely providing a probability that a specific file is malicious given a report and generating a ranked list of the most malicious items for an analyst to review. For the first instance, the analysts can evaluate the telemetry classifier results for any file based on the SHA1 hash. In addition, the analyst has the option of evaluating the results for reports where we have samples of the file which are labeled as malicious, labeled as benign, or not currently labeled. In the latter case, the status of the file is unknown and the telemetry classifier provides an indication whether or not the file is malicious. For files that are labeled by analysts or the signature detector as malicious, reports which are predicted to be benign should be considered potential false positive (FP) candidates; FPs are particularly problematic for anti-malware products. Similarly, files which are labeled by analysts as benign but the reports are predicted to be malicious are candidates for false negatives and can be analyzed further.

Next we investigate potential methods to defeat the proposed system. The main attack vector is to cause a report to be generated in such a way that the metadata mimics the features associated with benign files (i.e. a mimicry

attack), but in some cases, this is not an easy task to accomplish. For example, it would be very difficult to mimic the LS Hash of a benign file. Even if the attacker is able to some create malware with a LS Hash similar to a benign file, other features will help discriminate the malware from the legitimate file. For example, malware often tries to masquerade as legitimate software by copying legitimate signatures and certificate authorities. If the certificates are determined to be invalid by the AM engine, this provides a very strong hint to the telemetry classifier that the file is indeed malicious.

Another issue to consider is the accuracy of the reports analyzed in the previous section. For the results in Table 10, we consider all reports deemed malicious by signature detection to be generated by malware. Although relatively rare, false positives in the signatures lead to an increase in the number of reported detections. In this case, we will overestimate the effectiveness of the signature detection.

7 Related Work

The Microsoft Security Intelligence Report (SIR) [24] provides estimates of the number of detected malicious files for different malware families. The telemetry reports used to make the estimates in the SIR are the same as those used in this paper. As noted earlier, this report is based on known signature detections and does not attempt to estimate the amount of *unknown* malware.

Commercial software vendors have recently started using application and URL reputation to determine if an application or URL is malicious. For example, current versions of Symantec's security products [7], [8] and Microsoft's Internet Explorer [14] both employ telemetry reports to infer a file's reputation. The key observation is that as more users run an application or visit a URL, these entities can be considered more trustworthy. Applications which are only utilized by a few individuals are more likely to be malware. In this paper, we do not use the number of instances a particular SHA1 has been seen in the telemetry data so that we can try to detect zero-day attacks. Waiting some period of time to build a reputation could cause the system to miss many instances of a single polymorphic attack. An alternate version of our system could also be implemented with reputation data to better predict if an application is benign. In addition, earlier systems [7], [8] appear to utilize telemetry reports to build a reputation, but these papers do not attempt to classify the reports directly as proposed in this work.

Security researchers have written many papers on malware classification, and a recent survey of techniques used to detect malware is given in [16]. Most of the features used in the telemetry classifier are determined by static analysis of the file. As such, the telemetry classifier is closely related to early work in static malware classification of executable binaries. Schultz *et al.* [30] train classifiers to distinguish between malware and benign files based on three different feature sets (DLLs, strings, executable byte sequences). In [22], Kolter and Maloof train

several different classifiers based on n-grams of executable byte code sequences as features, among others.

There have been several in-depth malware analysis systems which have been developed over the years and could be utilized in Figure 1 including Anubis [17],[4], BitBlaze [32], and BAP [6]. Clustering and classification of the results of static analysis of files has been previously proposed for prefiltering for in-depth file analysis [18], [26], [34]. Our work differs in that we classify telemetry reports for files with analyzed on remote computers. Oberheide *et al.* [27] propose running simple clients on remote machines and transmitting the files to a backend service to be analyzed by a suite of commercial malware products. This work differs from the system described in this paper in that the entire file is not transmitted to the backend. Instead malware is detected by classifying the metadata in the telemetry reports.

8 Conclusions

For the first time, we estimate the total number of infected files including unknown files which have been predicted to be malicious using our telemetry report classifier. Based on a sample population of 50 million computers, we estimate that 93% of the files observed in the telemetry in October 2010 are malicious. While this estimate is somewhat biased, it confirms our suspicion that malware is a serious problem. We are somewhat encouraged that the current signatures have identified 82.8% of the known and predicted malware; we feel that this percentage could have been much worse, and the telemetry classifier allows us to measure our progress.

New AV signatures cannot be automatically generated using the proposed system: the false positive rate is too high. However we believe this telemetry classifier can serve several useful purposes including monitoring the current AV signature detection rates, automatically requesting samples, and ranking unknown files for more in-depth automated classification. The consequences of an FP are low: a user may be prompted to submit an unknown file which turns out to be benign or the in-depth analysis system spends a few minutes investigating a benign file. These outcomes can be minimized by only selecting files for analysis which are predicted to be malicious with a high probability.

Acknowledgments

We thank Misha Bilenko, Matthew Richardson, Ofer Dekel, and Galen Andrew for providing some of the machine learning algorithms used in this study. We also thank the anonymous reviewers for their insightful comments.

References

1. Andrew, G., Gao, J.: Scalable training of l1-regularized log-linear models. In: Proc. of the 24th International Conference on Machine Learning (ICML), Corvalis, OR. pp. 33–40. ACM, New York, NY (2007)

2. Bayer, U., Habibi, I., Balzarotti, D., Kirda, E., Kruegel, C.: A view on current malware behaviors. In: Proc. of 2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET), Boston, MA, USA (2009)
3. Bayer, U., Comparetti, P.M., Hlauschek, C., Kruegel, C., Kirda, E.: Scalable, behavior-based malware clustering. In: Proc. of the 16th Annual Network and Distributed System Security Symposium (NDSS), San Diego, CA (February 2009)
4. Bayer, U., Kruegel, C., Kirda, E.: TTAnalyze: A tool for analyzing malware. In: Proc. of 15th Annual Conference of the European Institute for Computer Antivirus Research (EICAR) (2006)
5. Bishop, C.: Pattern Recognition and Machine Learning. Springer (2006)
6. Brumley, D., Jager, I., Avgerinos, T., Schwartz, E.J.: Bap: Binary analysis platform. In: Proc. of the 2011 Conference on Computer Aided Verification (CAV) (2011)
7. C. Nachenberg, V. Seshadri, Z.R.: An analysis of real-world effectiveness of reputation-based security. In: Proc. of Virus Bulletin Conference (VB). pp. 178–183 (2010)
8. Chau, D.H., Nachenberg, C., Wilhelm, J., Wright, A., Faloutsos, C.: Polonium: Tera-scale graph mining and inference for malware detection. In: Proc. of SIAM International Conference on Data Mining (SDM) (2011)
9. Christodorescu, M., Jha, S., Kruegel, C.: Mining specifications of malicious behavior. In: Proc. of the 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE). pp. 5–14 (2007)
10. Edelman, B.: Adverse selection in online "trust" certifications. In: Fifth Workshop on the Economics of Information Security. pp. 26–28 (2006)
11. Freund, Y., Schapire, R.: Large margin classification using the perceptron algorithm. In: Machine Learning. pp. 277–296 (1999)
12. Friedman, J.: Greedy function approximation: a gradient boosting machine. In: Annals of Statistics. pp. 1189–1232 (2001)
13. Group, A.P.W.: Phishing activity trends report, 3rd quarter 2009 (2010), http://www.antiphishing.org/reports/apwg_report_Q3_2009.pdf
14. Haber, J.: Smartscreen application reputation in ie9 (2011), http://blogs.msdn.com/b/ie/archive/2011/05/17/smartscreen-174-application-reputation-in-ie9.aspx
15. Hu, W., Liao, Y., Vemuri, V.R.: Robust support vector machines for anomaly detection. In: Proc. 2003 International Conference on Machine Learning and Applications (ICMLA). pp. 23–24 (2003)
16. Idika, N., Mathur, A.: A survey of malware detection techniques. Tech. rep., Purdue Univ. (February 2007), http://www.eecs.umich.edu/techreports/cse/2007/CSE-TR-530-07.pdf
17. Iseclab: Anubis, analyzing unknown binaries. http://anubis.iseclab.org
18. Jacob, G., Comparetti, P.M., Neugschwandtner, M., Kruegel, C., Vigna, G.: A static, packer-agnostic filter to detect similar malware samples. In: Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA) (2012)
19. Jang, J., Brumley, D., Venkataraman, S.: Bitshred: feature hashing malware for scalable triage and semantic analysis. In: Proc. of the 18th ACM conference on Computer and communications security (CCS). pp. 309–320 (2011)
20. Jiang, X., Wang, X., Xu, D.: Stealthy malware detection through vmm-based "out-of-the-box" semantic view reconstruction. In: Proc. of the ACM Conference on Computer and Communications Security (CCS). pp. 128–138 (2007)

21. Kirda, E., Kruegel, C., Banks, G., Vigna, G., Kemmerer, R.A.: Behavior based spyware detection. In: Proc. of the 15th USENIX Security Symposium. pp. 273–288 (2006)
22. Kolter, J., Maloof, M.: Learning to detect and classify malicious executables in the wild. In: Journal of Machine Learning Research (JMLR). pp. 2721–2744 (2006)
23. Manning, C.D., Raghavan, P., Schtze, H.: An Introduction to Information Retrieval. Cambridge University Press (2009)
24. Microsoft: Microsoft security intelligence report, july - december 2010 (2011), `http://www.microsoft.com/security/sir/default.aspx`
25. Moser, A., Kruegel, C., Kirda, E.: Limits of static analysis for malware detection. In: Proc. of the 23rd Annual Computer Security Applications Conference (ACSAC). pp. 421–430 (2007)
26. Neugschwandtner, M., Comparetti, P.M., Jacob, G., Kruegel, C.: Forecast skimming off the malware cream. In: 27th Annual Computer Security Applications Conference (ACSAC) (2011)
27. Oberheide, J., Cooke, E., Jahanian, F.: Cloudav: N-version antivirus in the network cloud. In: Proc. of the 17th Conference on Security Symposium. pp. 91–106 (2008)
28. Perdisci, R., Lanzi, A., Lee, W.: Mcboost: Boosting scalability in malware collection and analysis using statistical classification of executables. In: Proc. of the 2008 Annual Computer Security Applications Conference (ACSAC). pp. 301–310 (2008)
29. Preda, M., Christodorescu, M., Jha, S., Debray, S.: A semantics-based approach to malware detection. In: Proc. of the 34th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages. pp. 377–388 (2007)
30. Schultz, M., Eskin, E., Zadok, E., Stolfo, S.: Data mining methods of detection of new malicious executables. In: Proc. of the 2001 IEEE Symposium on Security and Privacy (SP). pp. 38–49. IEEE Press, New York (2001)
31. Shalev-Shwartz, S., Singer, Y., Srebro, N.: Pegasos: Primal estimated sub-gradient solver for svm. In: Proc. of the 24th International Conference on Machine Learning (ICML), Corvalis, OR. pp. 807–814. ACM, New York, NY (2007)
32. Song, D., Brumley, D., Yin, H., Caballero, J., Jager, I., Kang, M.G., Liang, Z., Newsome, J., Poosankam, P., Saxena, P.: Bitblaze: A new approach to computer security via binary analysis. In: Proc. of the 4th International Conference on Information Systems Security (ICISS) (2008)
33. Stolfo, S., Wang, K., Li, W.: Towards stealthy malware detection. In: Christodorescu, M., Jha, S., Maughan, D., Song, D., Wang, C. (eds.) Malware Detection. Springer (2007)
34. Wicherski, G.: pehash: A novel approach to fast malware clustering. In: USENIX Workshop Large-Scale Exploits and Emergent Threats (LEET) (2009)
35. Zhang, B., Yin, J., Hao, J., Zhang, D., Wang, S.: Malicious codes detection based on ensemble learning. In: Proc. of Autonomic and Trusted Computing (ATC). pp. 468–477 (2007)
36. Zhang, J., Jin, R., Yang, Y., Hauptmann, A.G.: Modified logistic regression: An approximation to svm and its applications in large-scale text categorization. In: Proc. of the 20th International Conference on Machine Learning (ICML). Menlo Park. pp. 888–895 (2003)