Malware Detection using Statistical Analysis of Byte-Level File Content

S. Momina Tabish, M. Zubair Shafiq, Muddassar Farooq
Next Generation Intelligent Networks Research Center (nexGIN RC)
National University of Computer & Emerging Sciences (FAST-NUCES)
Islamabad, 44000, Pakistan
{momina.tabish,zubair.shafiq,muddassar.farooq}@nexginrc.org

ABSTRACT

Commercial anti-virus software are unable to provide protection against newly launched (a.k.a "zero-day") malware. In this paper, we propose a novel malware detection technique which is based on the analysis of byte-level file content. The novelty of our approach, compared with existing content based mining schemes, is that it does not memorize specific *byte-sequences* or *strings* appearing in the *actual* file content. Our technique is non-signature based and therefore has the potential to detect previously unknown and zero-day malware. We compute a wide range of statistical and information-theoretic features in a block-wise manner to quantify the byte-level file content. We leverage standard data mining algorithms to classify the file content of every block as *normal* or *potentially malicious*. Finally, we correlate the block-wise classification results of a given file to categorize it as *benign* or *malware*. Since the proposed scheme operates at the byte-level file content; therefore, it does not require any a priori information about the filetype. We have tested our proposed technique using a benign dataset comprising of six different filetypes — `DOC`, `EXE`, `JPG`, `MP3`, `PDF` and `ZIP` and a malware dataset comprising of six different malware types — backdoor, trojan, virus, worm, constructor and miscellaneous. We also perform a comparison with existing data mining based malware detection techniques. The results of our experiments show that the proposed non-signature based technique surpasses the existing techniques and achieves more than 90% detection accuracy.

Categories and Subject Descriptors

D.4.6 [**Security and Protection**]: Invasive Software

General Terms

Experimentation, Security

Keywords

Computer Malware, Data Mining, Forensics

CSI-KDD'09, June 28, 2009, Paris, France.

1. INTRODUCTION

Sophisticated malware is becoming a major threat to the usability, security and privacy of computer systems and networks worldwide [1], [2]. A wide range of host-based solutions have been proposed by researchers and a number of commercial anti-virus (AV) software are also available in the market [5]–[21]. These techniques can broadly be classified into two types: (1) static, and (2) dynamic. Static techniques mostly operate on machine-level code and disassembled instructions. In comparison, dynamic techniques mostly monitor the behavior of a program with the help of an API call sequence generated at run-time. The application of dynamic techniques in AV products is of limited use because of the large processing overheads incurred during run-time monitoring of API calls; as a result, the performance of computer systems significantly degrades. In comparison, the processing overhead is not a serious concern for static techniques because the scanning activity can be scheduled offline in an idle time. Moreover, static techniques can also be deployed as an *in-cloud network service* that moves complexity from an end-point to the network cloud [28].

Almost all static malware detection techniques including commercial AV software — either signature-, or heuristic-, or anomaly-based — use specific *content signatures* such as byte sequences and strings. A major problem with the content signatures is that they can easily be defeated by packing and basic code obfuscation techniques [3]. In fact, the majority of malware that appears today is a simple repacked version of old malware [4]. As a result, it effectively evades the *content signatures* of old malware stored in the database of commercial AV products. To conclude, existing commercial AV products cannot even detect a simple repacked version of previously detected malware.

The security community has expanded significant effort in application of data mining techniques to discover patterns in the malware content, which are not easily evaded by code obfuscation techniques. Two most well-known data mining based malware detection techniques are 'strings' (proposed by Schultz *et al* [7]) and 'KM' (proposed by Kolter *et al* [8]). We take these techniques as a benchmark for comparative study of our proposed scheme.

The novelty of our proposed technique — in contrast to the existing data mining based technique — is its purely non-signature paradigm: it does not remember exact file content/contents for malware detection. It is a static malware detection technique which should be, intuitively speaking, robust to the most commonly used evasion techniques. The proposed technique computes a diverse set of statistical

and information-theoretic features in a block-wise manner on the byte-level file content. The generated feature vector of every block is then given as an input to standard data mining algorithms (J48 decision trees) which classify the block as *normal* (`n`) or *potentially malicious* (`pm`). Finally, the classification results of all blocks are correlated to categorize the given file as *benign* (`B`) or *malware* (`M`). If a file is split into k equally-sized blocks $(b_1, b_2, b_3, \cdots, b_k)$ and n statistical features are computed for every k-th block $(f_{k,1}, f_{k,2}, f_{k,3}, \cdots, f_{k,n})$, then mathematically our scheme can be represented as:

$$\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \end{pmatrix} \overset{\mathbf{F}}{\Rightarrow} \begin{pmatrix} f_{1,1}, f_{1,2} \cdots f_{1,n} \\ f_{2,1}, f_{2,2} \cdots f_{2,n} \\ \vdots \\ f_{k,1}, f_{k,2} \cdots f_{k,n} \end{pmatrix} \overset{\mathbf{D}}{\Rightarrow} \begin{pmatrix} \texttt{n/pm} \\ \texttt{n/pm} \\ \vdots \\ \texttt{n/pm} \end{pmatrix} \overset{\mathbf{C}}{\Rightarrow} \texttt{B/M},$$

where (**F**) is a suitable feature set, (**D**) is a data mining algorithm for classification of individual blocks. The file is eventually categorized as *benign* (`B`) or *malware* (`M`) by the correlation module (**C**). Once a suitable feature set (**F**) and a data mining algorithm (**D**) are selected, we test the accuracy of the solution using a benign dataset consisting of six filetypes: `DOC`, `EXE`, `JPG`, `MP3`, `PDF` and `ZIP`; and a malware dataset comprising of six different malware types: backdoor, trojan, virus, worm, constructor and miscellaneous. The results of our experiments show that our scheme is able to provide more than 90% detection accuracy[1] for detecting malware, which is an encouraging outcome. To the best of our knowledge, this is the first pure non-signature based data mining malware detection technique using statistical analysis of the static byte-level file contents.

The rest of the paper is organized as follows. In Section 2 we provide a brief overview of related work in the domain of data mining malware detection techniques. We describe the detailed architecture of our malware detection technique in Section 3. We discuss the dataset in Section 4. We report the results of pilot studies in Section 5. In Section 6, we discuss the knowledge discovery process of the proposed technique by visualizing the learning models of data mining algorithms used for classification. Finally, we conclude the paper with an outlook to our future work.

2. RELATED WORK

In this section we explain the details of most relevant malware detection techniques: (1) 'strings' by Schultz *et al* [7], (2) 'KM' by Kolter *et al* [8] and (3) 'NG' by Stolfo *et al* [16], [17]. In our comparative study, we use 'strings' and 'KM' as benchmarks for comparison, whereas 'NG' effectively uses just one of the many statistical features used in our proposed technique.

2.1 Schultz *et al* — Strings

In [7], Schultz *et al* use several data mining techniques to distinguish between the benign and malicious executables in Windows or MS-DOS format. They have done experiments on a dataset that consists of 1,001 benign and 3,265 malicious executables. These executables have 206 benign and 38 malicious samples in the portable executable (PE) file format. They have collected most of the benign executables from Windows 98 systems. They use three different approaches to statically extract features from executables.

The first approach extracts DLL information inside PE executables. Further, the DLL information is extracted using three types of feature vectors: (1) the list of DLLs (30 boolean values), (2) the list of DLL function calls (2,229 boolean values), and (3) the number of different function calls within each DLL (30 integer values). RIPPER — an inductive rule-learning algorithm — is used on top of every feature vector for classification. These schemes based on DLL information provides an overall detection accuracy of 83.62%, 88.36% and 89.07% respectively. Enough details about the DLLs are not provided, so we could not implement this scheme in our study.

The second feature extraction approach extracts strings from the executables using GNU *strings* program. Naïve Bayes classifier is used on top of extracted strings for malware detection. This scheme provides an overall detection accuracy of 97.11%. This scheme is reported to give the best results amongst all, and we have implemented it for our comparative study.

The third feature extraction approach uses byte sequences (n-grams) using hexdump. The authors do not explicitly specify the value of n used in their study. However, from an example provided in the paper, we deduce it to be 2 (bi-grams). The Multi-Naïve Bayes algorithm is used for classification. This algorithm uses voting by a collection of individual Naïve Bayes instances. This scheme provides an overall detection accuracy of 96.88%.

The results of their experiments reveal that Naïve Bayes algorithm with strings is the most effective approach for detecting the unseen malicious executables with reasonable processing overheads. The authors acknowledge the fact that the string features are not robust and can be easily defeated. Multi-Naïve Bayes with byte sequences also provides a relatively high detection accuracy, however, it has large processing and memory requirements. Byte sequence technique was later improved by Kolter *et al* and is explained below.

2.2 Kolter *et al* — KM

In [8], Kolter *et al* use n-gram analysis and data mining approaches to detect malicious executables in the wild. They use n-gram analysis to extract features from 1,971 benign and 1,651 malicious PE files. The PE files have been collected from machines running Windows 2000 and XP operating systems. The malicious PE files are taken from an older version of the VX Heavens Virus Collection [27].

The authors evaluate their approach for two classification problems: (1) classification between the benign and malicious executables, and (2) categorization of executables as a function of their payload. The authors have categorized only three types — mailer, backdoor and virus — due to the limited number of malware samples.

Top n-grams with the highest information gain are taken as binary features (`T` if present and `F` if absent) for every PE file. The authors have done pilot studies to determine the size of n-grams, the size of words and the number of top n-grams to be selected as features. A smaller dataset

[1]Throughout this text, the terms *detection accuracy* and *Area Under ROC Curve (AUC)* are used interchangeably. The AUC ($0 \leq \text{AUC} \leq 1$) is used as a yardstick to determine the detection accuracy. Higher values of AUC mean high *true positive* (tp) rate and low *false positive* (fp) rate [30]. At AUC = 1, *tp* rate = 1 and *fp* rate = 0.

consisting of 561 benign and 476 malicious executables is considered in this study. They have used 4-grams, one byte word and top 500 n-grams as features.

Several inductive learning methods, namely instance-based learner, Naïve Bayes, support vector machines, decision trees and boosted versions of instance-based learner, Naïve Bayes, support vector machines and decision trees are used for classification. The same features are provided as input to all classifiers. They report the detecting accuracy as the area under an ROC curve (AUC) which is a more complete measure compared with the detection accuracy [29]. AUCs show that the boosted decision trees outperform rest of the classifiers for both classification problems.

2.3 Stolfo *et al* — NG

In a seminal work, Stolfo *et al* have used n-gram analysis for filetype identification [15] and later for malware detection [16], [17]. Their earlier work, called fileprint analysis, uses 1-gram byte distribution of a whole file and compares it with different filetype models to determine the filetype. In their later work, they detect malware embedded in `DOC` and `PDF` files using three different models single centroid, multi centroid, and exemplar of benign byte distribution of the whole files. A distance measure, called Mahanalobis Distance, is calculated between the n-gram distribution of these models and a given test file. They also use 1-gram and 2-gram distributions to test their approach on a dataset comprising of 31 benign application executables, 331 benign executables from the System32 folder and 571 viruses. The experimental results have shown that their proposed technique is able to detect a considerable proportion of the malicious files. However, their proposed technique is specific to embedded malware and does not deal with detection of stand-alone malware.

3. ARCHITECTURE OF PROPOSED TECHNIQUE

In this section, we explain our data mining based malware detection. It is important to emphasize that our technique does not require any a priori information about the filetype of a given file; as a result, our scheme is robust to subtle file header obfuscations crafted by an attacker. Moreover, our technique is also able to classify the malware as a function of their payload, i.e. it can detect the family of a given malware.

The architecture of proposed technique is shown in Figure 1. It consists of four modules: (1) block generator (**B**), (2) feature extractor (**F**), (3) data miner (**D**), and (4) correlation (**C**). It can accept any type of file as an input. We now discuss the details of every module.

3.1 Block Generator Module (B)

The block generator module divides the byte-level contents of a given file into fixed-sized chunks — known as blocks. We have used blocks for reducing the processing overhead of the module. In future, we want to analyze the benefit of using variable-sized blocks as well. Remember that using a suitable block size plays a critical role in defining the accuracy of our framework because it puts a lower limit on the minimum size of malware that our framework can detect. We have to compromise a trade-off between the amount of available information per block and the accuracy of the system. In this study, we have set the block size to 1024 bytes ($= 1K$). For instance, if the file size is $100K$ bytes then the file is split into 100 blocks. The frequency histograms for 1-, 2-, 3-, and 4-gram of byte values are calculated for each block. These histograms are given as input to the feature extraction module.

3.2 Feature Extraction Module (F)

Feature extraction module computes a number of statistical and information-theoretic features on the histograms for each block generated by the previous module. Overall, the features' set consist of 13 diverse features, which are separately computed on 1, 2, 3, and 4 gram frequency histograms.[2] This brings the total size of features' set to 52 features per block. We now provide brief descriptions of every feature.

3.2.1 Simpson's Index

The Simpson's index [31] is a measure defined in an ecosystem, which quantifies the diversity of species in a habitat. It is calculated using the following equation:

$$S_i = \frac{\sum n(n-1)}{N(N-1)}, \tag{1}$$

where n is the frequency of byte-values of consecutive n-grams and N is the total number of bytes in a block i.e, 1000 in our case. A value of zero shows no significant difference between frequency of n-grams in a block. Similarly, as the value of S_i increases, the variance in frequency of n-grams in a block also increases.

In all subsequent feature definitions, we represent with X_i the frequency of j^{th} n-gram in i^{th} block, where j is varying from $0-255$ for 1-gram, $0-65535$ for 2-grams, $0-16777215$ for 3-grams, and $0-4294967295$ for 4-grams.

3.2.2 Canberra Distance

This distance measures the sum of series of fractional differences between coordinates of a pair of objects [31]. Mathematically we represent it as:

$$CA(i) = \sum_{j=0}^{n} \frac{\mid X_j - X_{j+1} \mid}{\mid X_j \mid + \mid X_{j+1} \mid} \tag{2}$$

3.2.3 Minkowski Distance of Order

This is a generalized metric to measure the differences between absolute magnitude of differences between a pair of objects:

$$m_i = \sqrt[\lambda]{\sum_{j=0}^{n} \mid X_j - X_{j+1}, \mid^{\lambda}} \tag{3}$$

where we have used $\lambda = 3$ as suggested in [31].

3.2.4 Manhattan Distance

It is a special case of Minkowski distance [31] with $\lambda = 1$.

$$mh_i = \sum_{j=0}^{n} \mid X_j - X_{j+1} \mid \tag{4}$$

[2]In rest of the paper, we use the generic term n-grams once we want to refer to all 4-grams separately.

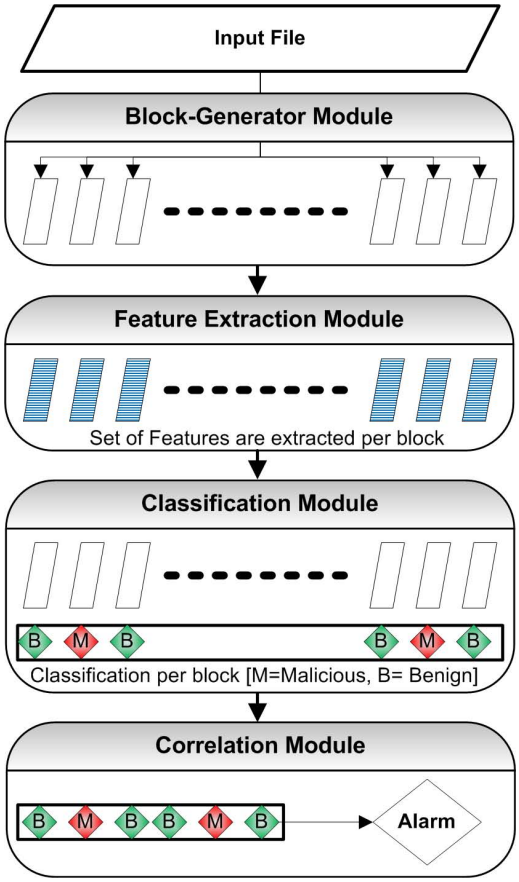

Figure 1: Architecture of our proposed technique

3.2.5 Chebyshev Distance

Chebyshev distance measure is also called maximum value distance. It measures the absolute magnitude of differences between coordinates of a pair of objects:

$$ch_i = max_j \mid X_j - X_{j+1} \mid \tag{5}$$

It is a special case of Minkowski difference [31] with $\lambda = \infty$.

3.2.6 Bray Curtis Distance

It is a normalized distance measure which is defined as the ratio of absolute difference of frequencies of n-grams and the sum of their frequencies [31]:

$$bc_i = \frac{\sum_{j=0}^{n} \mid X_j - X_{j+1} \mid}{\sum_{j=0}^{n} (X_j + X_{j+1})} \tag{6}$$

3.2.7 Angular Separation

This feature models the similarity of two vectors by taking cosine of the angle between them [31]. A higher value of angular separation between two vectors shows that they are similar.

$$AS_i = \frac{\sum_{j=0}^{n} X_j \cdot X_{j+1}}{(\sum_{j=0}^{n} X_j^2 \cdot \sum_{j=0}^{n} X_{j+1}^2)^{1/2}} \tag{7}$$

3.2.8 Correlation Coefficient

The standard angular separation between two vectors which is centered around the mean of the its magnitude values is called correlation coefficient [31]. This again measures the similarity between two values:

$$CC_i = \frac{\sum_{j=0}^{n} (X_j - \overline{X}_i) \cdot (X_{j+1} - \overline{X}_i)}{(\sum_{j=0}^{n} (X_j - \overline{X}_i)^2 \cdot \sum_{j=0}^{n} (X_{j+1} - \overline{X}_i)^2)^{1/2}}, \tag{8}$$

where $\overline{X}_i$ is the mean of frequencies of n-grams in a given block i.

3.2.9 Entropy

Entropy measures the degree of dispersal or concentration of a distribution. In information-theoretic terms, entropy of a probability distribution defines the minimum average number of bits that a source requires to transmit symbols according to that distribution [31]. Let R be a discrete random variable such that $R = \{r_i, i \in \Delta_n\}$, where Δ_n is the image of a random variable. Then entropy of R is defined as:

$$E(R) = -\sum_{i \in \Delta_n} t(r_i) \log_2 t(r_i), \tag{9}$$

where $t(r_i)$ is the frequency of n-grams in a given block.

3.2.10 Kullback - Leibler Divergence

KL Divergence is a measure of the difference between two probability distributions [31]. It is often referred to as a distance measure between two distributions. Mathematically, it is represented as:

$$KL_i(X_j \mid\mid X_{j+1}) = \sum_{j=0}^{n} X_j log \frac{X_j}{X_{j+1}} \quad (10)$$

3.2.11 Jensen-Shannon Divergence

It is a popular measure in probability theory and statistics and measures the similarity between two probability distributions [31]. It is also known as Information Radius (IRad). Mathematically, it is represented as:

$$JSD_i(X_j \mid\mid X_{j+1}) = \frac{1}{2} D(X_j \mid\mid M) + \frac{1}{2} D(X_{j+1} \mid\mid M) \quad (11)$$

where $M = \frac{1}{2}(X_j + X_{j+1})$.

3.2.12 Itakura-Saito Divergence

Itakura-Saito Divergence is a special form of Bregman distance [31].

$$B_F(X_j, X_{j+1}) = \sum_{j=0}^{n} (\frac{X_j}{X_{j+1}} - log \frac{X_j}{X_{j+1}} - 1) \quad (12)$$

which is generated by the convex function $F_i(X_j) = -\sum \log X_j$.

3.2.13 Total Variation

It measures the largest possible difference between two probability distributions X_j and X_{j+1} [31]. It is defined as:

$$\delta_i(X_j, X_{j+1}) = \frac{1}{2} \sum_j \mid X_j - X_{j+1} \mid \quad (13)$$

3.3 Data Mining based Classification Module (D)

The classification module gets as an input the feature vector in the form of an `arff` file [26]. This feature file is then further presented for classification to 6 sub-modules. The six modules actually contain learnt models of six types of malicious files: backdoor, virus, worm, trojan, constructor and miscellaneous. The feature vector file is presented to all sub-modules in parallel and they produce an output of `n` or `pm` per block. In addition to this, the output of the classification sub-modules provide us insights into the payload of the malicious file.

Boosted decision tree is used for classifying each block as `n` or `pm`. We have used AdaBoostM1 algorithm for boosting decision tree (J48) [24]. We have selected this classifier after extensive pilot studies which are detailed in Section 5. We provide brief explanations of decision tree (J48) and boosting algorithm (AdaBoostM1) below.

3.3.1 Decision Tree (J48)

We have used C4.5 decision tree (J48) implemented in Waikato Environment for Knowledge Acquisition (WEKA) [26]. It uses the concept of information entropy to build the tree. Every feature is used to split the dataset into smaller subsets and the normalized information gain is calculated. The feature with highest information gain is selected for decision making.

Table 1: Statistics of benign files used in this study

Filetype	Qty.	Avg. Size (kilo-bytes)	Min. Size (kilo-bytes)	Max. Size (kilo-bytes)
DOC	300	1,015.2	44	7,706
EXE	300	4,095.0	48	15,005
JPG	300	1,097.8	3	1,629
MP3	300	3,384.4	654	6,210
PDF	300	1,513.1	25	20,188
ZIP	300	1,489.6	8	9,860

3.3.2 Boosting (AdaBoostM1)

We have used AdaBoostM1 algorithm implemented in WEKA [24]. As the name suggests, it is a meta algorithm which is designed to improve the performance of base learning algorithms. AdaBoostM1 keeps calling the weak classifier until a pre-defined number of times and tweaks those instances that have resulted in misclassification. In this way, it keeps on adapting itself with the ongoing classification process. It is known to be sensitive to outliers and noisy data.

3.4 Correlation Module (C)

The correlation module gets the per block classification results in the form of `n` or `pm`. It then calculates the correlation among the blocks which are labeled as `n` or `pm`. Depending upon the fraction of `n` and `pm` blocks in a file, the file is classified as malicious or benign. We can also set a threshold for tweaking the final classification decision. For instance if we set the threshold to 0.5, a file having 4 benign and 6 malicious blocks will be classified as malicious, and vice-versa.

4. DATASET

In this section, we present an overview of the dataset used in our study. We first describe the benign and then the malware dataset used in our experiments. It is important to note that in this work we are not differentiating between packed and non-packed files. Our scheme works regardless of the packed/non-packed nature of the file.

4.1 Benign Dataset

The benign dataset for our experiments consists of six different filetypes: `DOC`, `EXE`, `JPG`, `MP3`, `PDF` and `ZIP`. These filetypes encompass a broad spectrum of commonly used files ranging from compressed to redundant and from executables to document files. Each set of benign files contains 300 typical samples of the corresponding filetype, which provide us with 1,800 benign files in total. We have ensured the generality of the benign dataset by randomizing the sample sources. More specifically, we queried well-known search engines with random keywords to collect these files. In addition, typical samples are also collected from the local network of our virology lab.

Some pertinent statistics of the benign dataset used in this study are tabulated in Table 1. It can be observed from Table 1 that the benign files have very diverse sizes varying from 3 KB to 20 MB, with an average file size of approximately 2 MB. The divergence in sizes of benign files is important as malicious programs are inherently smaller in size for ease of propagation.

Table 2: Statistics of malware used in this study

Maj. Category	Qty.	Avg. Size (kilo-bytes)	Min. Size (bytes)	Max. Size (kilo-bytes)
Backdoor	3,444	285.6	56	9,502
Constructor	172	398.5	371	5,971
Trojan	3114	135.7	12	4,014
Virus	1048	50.7	175	1,332
Worm	1471	72.3	44	2,733
Miscellaneous	1062	197.7	371	14,692

4.2 Malware Dataset

We have used 'VX Heavens Virus Collection' [27] database which is available for free download in the public domain. Malware samples, especially recent ones, are not easily available on the Internet. Computer security corporations do have an extensive malware collection, but unfortunately they do not share their malware databases for research purposes. This is a comprehensive database that contains a total of 37,420 malware samples. The samples consist of backdoors, constructors, flooders, bots, nukers, sniffers, droppers, spyware, viruses, worms and trojans etc. We only consider `Win32` based malware in PE file format. The filtered dataset used in this study contains 10,311 `Win32` malware samples. To make our study more comprehensive, we divide the malicious executables based on the function of their payload. The malicious executables are divided into six major categories such as backdoor, trojan, virus, worm, constructor, and miscellaneous (malware like nuker, flooder, virtool, hacktool etc). We now provide a brief explanation of each of the six malware categories.

4.2.1 Backdoor

A backdoor is a program which allows bypassing of standard authentication methods of an operating system. As a result, remote access to computer systems is possible without explicit consent of the users. Information logging and sniffing activities are possible using the gained remote access.

4.2.2 Constructor

This category of malware mostly includes toolkits for automatically creating new malware by varying a given set of input parameters.

4.2.3 Worm

The malware in this category spreads over the network by replicating itself.

4.2.4 Trojan

A trojan is a broad term that refers to stand alone programs which appear to perform a legitimate function but covertly do possibly harmful activities such as providing remote access, data destruction and corruption.

4.2.5 Virus

A virus is a program that can replicate itself and attach itself with other benign programs. It is probably the most well-known type of malware and has different flavors.

4.2.6 Miscellaneous

The malware in this category include DoS (denial of service), nuker, exploit, hacktool and flooders. The DoS and nuker based malware allow an attacker to launch malicious activities at a victim's computer system that can possibly result in a denial of service attack. These activities can result in slow down, restart, crash or shutdown of a computer system. Exploit and hacktool exploit vulnerabilities in a system's implementation which most commonly results in buffer overflows. Flooder initiates unwanted information floods such as email, instant messaging and SMS floods.

The detailed statistics of the malware used in our study is provided in Table 2. The average malware size in this dataset is 64.2 KB. The sizes of malware samples used in our study vary from 4 bytes to more than 14 MB. Intuitively speaking, small sized malware are harder to detect than the larger ones.

Table 3: AUC's for detecting malicious executables vs benign files. Bold entries in every column represent the best results.

Classifier	Back	Cons	Misc	Troj	Virus	Worm
Strings						
RIPPER	0.591	0.611	0.690	0.685	0.896	0.599
NB	0.615	0.610	0.714	0.751	0.944	0.557
M-NB	0.615	0.606	0.711	0.755	**0.952**	0.575
B-J48	**0.625**	**0.652**	**0.765**	**0.762**	0.946	**0.642**
KM						
SMO	0.720	–	0.611	0.755	0.865	0.750
B-SMO	0.711	–	0.611	0.766	0.931	0.759
NB	0.715	–	0.610	0.847	**0.947**	0.750
B-NB	0.715	–	0.606	0.821	0.939	0.760
J48	0.712	–	0.560	0.805	0.850	0.817
B-J48	**0.795**	–	**0.652**	**0.851**	0.921	0.820
IBk	0.752	–	0.611	0.850	0.942	**0.841**
Proposed solution with 4 features						
J48	0.835	0.812	0.863	0.837	0.909	0.880
B-J48	**0.849**	**0.841**	**0.883**	**0.839**	**0.933**	**0.884**
NB	0.709	0.716	0.796	0.748	0.831	0.715
B-NB	0.709	0.722	0.794	0.746	0.807	0.707
IBk	0.779	0.817	0.844	0.791	0.917	0.812
B-IBk	0.782	0.817	0.844	0.791	0.918	0.812
Proposed solution with 52 features						
B-J48	**0.979**	**0.965**	**0.950**	**0.985**	**0.970**	**0.932**

5. PILOT STUDIES

In our initial set of experiments, we have conducted an extensive search in the design space to select the best features' set and classifier for our scheme. The experiments are done on 5% of total dataset to have small design cycle for our approach. Recall that in Section 3, we have introduced 13 different statistical and information-theoretic features. The pilot studies are aimed to convince ourselves that we need all of them. Moreover, we have also evaluated well-known data mining algorithms on our dataset in order to find the best classification algorithm for our problem. We have used Instance based (IBk) [22], Decision Trees (J48) [25], Naïve Bayes [23] and the boosted versions of these classifiers in our pilot study. For boosting we have used AdaBoostM1 algorithm [24]. We have utilized implementations of these algorithms available in WEKA [26].

5.1 Discussion on Pilot Studies

The classification results of our experiments are tabulated in Table 3 which show that the boosted Decision Tree (J48) significantly outperforms other classifiers in terms of detec-

Table 4: Feature Analysis. AUC's for detecting virus executables vs benign files using boosted J48. F1,F2, F3 and F4 correspond to 4 different features.

Feature	No. of Gram/feature	AUC
F1	1	0.823
F2	1	0.839
F3	1	0.866
F4	2	0.891
F1-F2	1, 1	0.940
F1-F3	1, 1	0.928
F1-F4	1, 2	0.932
F2-F4	1, 2	0.929
F1-F2-F4	1, 1, 2	**0.962**
F3-F2	1, 1	0.954
F3-F4	1, 2	0.913
F1-F2-F3-F4	1, 1, 1, 2	0.956

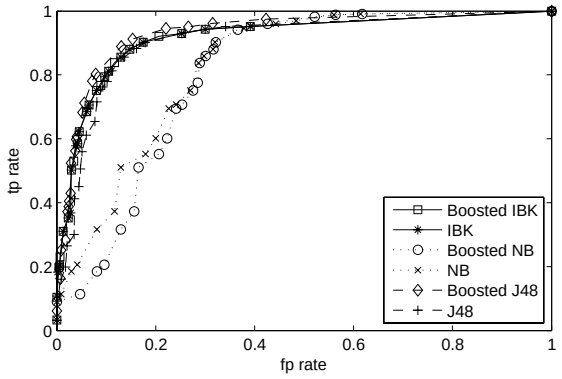

Figure 2: ROC plot for virus-benign classification

tion accuracy. Similarly we also evaluate the role of number of features and the number of n-grams on the accuracy of proposed approach. In Table 3, we first use four features namely: Simpson's index (F1), Entropy Rate (F2), Canberra distance (F3) on 1-gram and Simpson's index on 2-grams (F4). Our scheme achieves a significantly higher detection accuracy compared with strings and KM. We then tried different combinations of features and n-grams and tabulated the results in Table 4. It is obvious from Table 4 that once we move from a single feature from (F1) on 1-gram to (F4) on 2-grams the detection accuracy improves from 0.82 to 0.891. Once we use combination of features computed on 1-gram and 2-grams the accuracy approaches to 0.962 (see F1-F2-F4). This provided us the motivation to use all 13 features on 1-, 2-, 3- and 4-grams resulting in a total of 52 features. The ROC curve for the virus-benign classification using F1, F2, F3 and F4 features is shown in Figure 2.

It is clear in Table 3 that strings approach has a significantly higher accuracy for virus types compared with other malware types. We analyzed the reason behind this by looking at the signatures used by this method. We observed that typically viruses carry some common strings like "Chinese Hacker" and for strings approach they become its signatures. Since similar strings do not appear in other malware types; therefore, strings accuracy degrades to as low as 0.62 in case of backdoors.

Recall that KM uses 4-grams as binary features. KM follows the same pattern of higher accuracy for detecting viruses and relatively lower accuracy for other malware types. However, its accuracy results are significantly higher compared with the strings approach. Note that our proposed solution with 52 features not only provides the best detection accuracy for virus category but its accuracy remains consistent across all malware types. This shows the strength of using diverse features' set with a boosted J48 classifier.

6. RESULTS & DISCUSSION

Remember that our approach is designed to achieve a challenging objective: to distinguish between malicious files of type backdoor, virus, worm, trojan, constructor from benign files of types `DOC`, `EXE`, `JPG`, `MP3`, `PDF` and `ZIP` just on the basis of byte-level information.

We now explain our final experimental setup. The proposed scheme, as explained in Section 3.2, computes features on n-grams of each block of a given file. We create the training samples by randomly selecting 50 malicious and benign files each from malware and benign datasets respectively. We create six separate training samples for each malware category. We use these samples to train boosted decision tree and consequently get 6 classification models for each malware type.

For an easier understanding of the classification process, we take backdoor as an example. We have selected 50 backdoor files and 50 benign files to get a training sample for classifying backdoor and benign files. We train the data mining algorithm with this sample and as a result get the training model to classify a backdoor. Moreover, we further test, using this model, all six benign filetypes and backdoor malware files. It is important to note that this model is specifically designed to distinguish backdoor files from six benign filetypes (one-vs-all classification), where only 50 backdoor samples are taken in training. This number is considerably small keeping in view the problem at hand. Nonetheless, the backdoor classification using a single classifier completes in seven runs: six for benign filetypes and one for itself. In a similar fashion, one can classify malware types i.e., virus, worm, trojan, constructor and miscellaneous. Once our solution is trained for each category of malware, we test it on the benign dataset of 1800 files and the malware dataset of 10,311 files from VX Heavens. We ensure that the benign and malware files used in the training phase are not included in the testing phase to verify our claim of zero-day malware detection. The classification results are shown in Figure 3 in the form of AUCs. It is interesting to see that viruses, as expected, are easily classified by our scheme. In comparison trojans are programs that look and behave like benign programs, but perform some illegitimate activities. As expected, the classification accuracy for trojans is 0.88 — significantly smaller compared with viruses.

In order to get a better understanding, we have plotted the ROC graphs of classification results for each malware type in Figure 3. The classification results show that malware files are quite distinct from benign files in terms of byte-level file content. The ROC plot further confirms that virus are easily classified compared with other malware types while trojan and backdoors are relatively difficult to classify. Table 5 shows portions of the developed decision trees for all malware categories. As decision trees provide a simple and robust method of classification for a large dataset, it is interesting to note that malicious files are inherently different from the benign ones even at the byte-level.

Table 5: Portions of developed decision trees

```
|   |   |   |   KL1 > 0.022975
|   |   |   |   |   Entropy2 <= 6.822176
|   |   |   |   |   |   Manhattan1 <= 0.005411
|   |   |   |   |   |   |   Manhattan4 <= 0.000062: Malicious (20.0)
|   |   |   |   |   |   |   Manhattan4 > 0.000062: Benign (8.0/2.0)
|   |   |   |   |   | Manhattan1 > 0.005411: Malicious (538.0/6.0)
```

(a) between Backdoor and Benign files

```
|   |   CorelationCoefficient1 > 0.619523
|   |   |   Chebyshev4 <=1.405
|   |   |   |   Itakura2 <= 87.231983: Malicious (352.0/9.0)
|   |   |   |   Itakura2 > 87.231983
|   |   |   |   | TotalVariation1 <= 0.3415: Malicious (11.0)
|   |   |   |   | TotalVariation1 > 0.3415: Benign (8.0)
```

(b) between Trojan and Benign files

```
|   |   CorelationCoefficient4 > 0.187794
|   |   |   Simpson_Index_1 <= 0.005703
|   |   |   |   CorelationCoefficient3 <= 0.17584: Malicious (32.0)
|   |   |   |   CorelationCoefficient3 > 0.17584
|   |   |   |   |   Entropy1 <= 4.969689: Malicious (4.0/1.0)
|   |   |   |   |   Entropy1 > 4.969689: Benign (5.0)
|   |   |   Simpson_Index_1 > 0.005703: Benign (11.0)
```

(c) between Virus and Benign files

```
|   |   |   Entropy2 > 3.00231
|   |   |   |   Canberra2 <= 49.348481
|   |   |   |   |   Canberra1 <= 14.567909: Malicious (161.0)
|   |   |   |   |   Canberra1 > 14.567909
|   |   |   |   |   |   Itakura3 <= 126.178699: Malicious (5.0)
|   |   |   |   |   |   Itakura3 > 126.178699: Benign (5.0)
|   |   |   |   Canberra2 > 49.348481: Benign (13.0/1.0)
```

(d) between Worm and Benign files

```
CorelationCoefficient3 <= -0.013832
|   Itakura2 <= 5.905754
|   |   Itakura3 <= 5.208592
|   |   |   CorelationCoefficient4 <= -0.155078: Malicious (7.0)
|   |   |   CorelationCoefficient4 > -0.155078: Benign (43.0/6.0)
|   |   Itakura3 > 5.208592: Benign (303.0/5.0)
```

(e) between Constructor and Benign files

```
|   Entropy4 > 6.754364
|   |   KL1 <= 0.698772
|   |   |   Manhattan3 <= 0.001003
|   |   |   |   Entropy2 <= 6.411063: Malicious (29.0)
|   |   |   |   Entropy2 > 6.411063
|   |   |   |   |   KL1 <= 0.333918: Malicious (2.0)
|   |   |   |   |   KL1 > 0.333918: Benign (2.0)
|   |   |   Manhattan3 > 0.001003: Benign (3.0)
```

(f) between Miscellaneous and Benign files

The novelty of our scheme lies in the way the selected features have been computed — per block n-gram analysis, and the correlation between the blocks classified as benign or potentially malicious. The features used in our study are taken from statistics and information theory. Many of these features have already been used by researchers in other fields for similar classification problems. The chosen set of features is not, by any means, the optimal collection. The selection of optimal number of features remains an interesting problem which we plan to explore in our future work. Moreover, the executable dataset used in our study contained both packed and non-packed PE files. We plan to evaluate the robustness of our proposed technique on manually crafted packed file dataset.

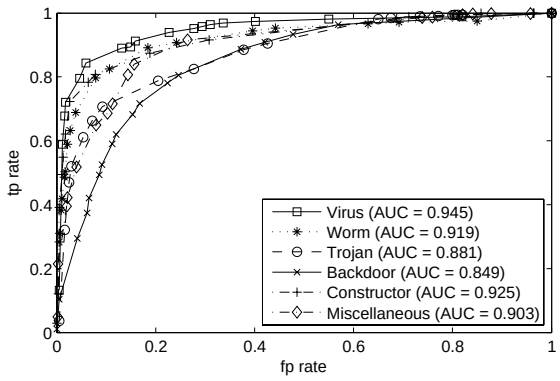

Figure 3: ROC plot for detecting malware from benign files

7. CONCLUSION & FUTURE WORK

In this paper we have proposed a non-signature based technique which analyzes the byte-level file content. We argue that such a technique provides implicit robustness against common obfuscation techniques — especially repacked malware to obfuscate signatures. *An outcome of our research is that malicious and benign files are inherently different even at the byte-level.*

The proposed scheme uses a rich features' set of 13 different statistical and information-theoretic features computed on 1-, 2-, 3- and 4-grams of each block of a file. Once we have calculated our features' set, we give it as an input to the boosted decision tree (J48) classifier. The choice of features' set and classifier is an outcome of extensive pilot studies done to explore the design space. The pilot studies demonstrate the benefit of our approach compared with other well-known data mining techniques: strings and KM approach. We have tested our solution on an extensive executable dataset. The results of our experiments show that our technique achieves 90% detection accuracy for different malware types. Another important feature of our framework is that it can also classify the family of a given malware file i.e. virus, trojan etc.

In future, we would like to evaluate our scheme on a larger dataset of benign and malicious executables and reverse engineer the features' set for further improving the detection accuracy. Moreover, we plan to evaluate the robustness of our proposed technique on a customized dataset containing manually packed executable files.

Acknowledgments

This work is supported by the National ICT R&D Fund, Ministry of Information Technology, Government of Pakistan. The information, data, comments, and views detailed herein may not necessarily reflect the endorsements of views of the National ICT R&D Fund.

We acknowledge M.A. Maloof and J.Z. Kolter for their valuable feedback regarding the implementation of strings and KM approaches. Their comments were of great help in establishing the experimental testbed used in our study. We also acknowledge the anonymous reviewers for their valuable suggestions pertaining to possible extensions of our study.

8. REFERENCES

[1] Symantec Internet Security Threat Reports I-XI (Jan 2002—Jan 2008).

[2] F-Secure Corporation, "F-Secure Reports Amount of Malware Grew by 100% during 2007", Press release, 2007.

[3] A. Stepan, "Improving Proactive Detection of Packed Malware", Virus Buletin, March 2006, available at `http://www.virusbtn.com/virusbulletin/archive/2006/03/vb200603-packed.dkb`

[4] R. Perdisci, A. Lanzi, W. Lee, "Classification of Packed Executables for Accurate Computer Virus Detection", Pattern Recognition Letters, 29(14), pp. 1941-1946, Elsevier, 2008.

[5] AVG Free Antivirus, available at `http://free.avg.com/`.

[6] Panda Antivirus, available at `http://www.pandasecurity.com/`.

[7] M.G. Schultz, E. Eskin, E. Zadok, S.J. Stolfo, "Data mining methods for detection of new malicious executables", IEEE Symposium on Security and Privacy, pp. 38-49, USA, IEEe Press, 2001.

[8] J.Z. Kolter, M.A. Maloof, "Learning to detect malicious executables in the wild", ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 470-478, USA, 2004.

[9] J. Kephart, G. Sorkin, W. Arnold, D. Chess, G. Tesauro, S. White, "Biologically inspired defenses against computer viruses", International Joint Conference on Artificial Intelligence (IJCAI), pp. 985-996, USA, 1995.

[10] R.W. Lo, K.N. Levitt, R.A. Olsson, "MCF: A malicious code filter", Computers & Security, 14(6):541-566, Elseveir, 1995.

[11] O. Henchiri, N. Japkowicz, "A Feature Selection and Evaluation Scheme for Computer Virus Detection", IEEE International Conference on Data Mining (ICDM), pp. 891-895, USA, IEEE Press, 2006.

[12] P. Kierski, M. Okoniewski, P. Gawrysiak, "Automatic Classification of Executable Code for Computer Virus Detection", International Conference on Intelligent Information Systems, pp. 277-284, Springer, Poland, 2003.

[13] T. Abou-Assaleh, N. Cercone, V. Keselj, R. Sweidan. "Detection of New Malicious Code Using N-grams Signatures", International Conference on Intelligent Information Systems, pp. 193-196, Springer, Poland, 2003.

[14] J.H. Wang, P.S. Deng, "Virus Detection using Data Mining Techniques", IEEE International Carnahan Conference on Security Technology, pp. 71-76, IEEE Press, 2003.

[15] W.J. Li, K. Wang, S.J. Stolfo, B. Herzog, "Fileprints: identifying filetypes by n-gram analysis", IEEE Information Assurance Workshop, USA, IEEE Press, 2005.

[16] S.J. Stolfo, K. Wang, W.J. Li, "Towards Stealthy Malware Detection", Advances in Information Security, Vol. 27, pp. 231-249, Springer, USA, 2007.

[17] W.J. Li, S.J. Stolfo, A. Stavrou, E. Androulaki, A.D. Keromytis, "A Study of Malcode-Bearing Documents", International Conference on Detection of Intrusions & Malware, and Vulnerability Assessment (DIMVA), pp. 231-250, Springer, Switzerland, 2007.

[18] M.Z. Shafiq, S.A. Khayam, M. Farooq, "Embedded Malware Detection using Markov n-Grams", International Conference on Detection of Intrusions & Malware, and Vulnerability Assessment (DIMVA), pp. 88-107, Springer, France, 2008.

[19] M. Christodorescu, S. Jha, and C. Kruegal, "Mining Specifications of Malicious Behavior", European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2007), pp. 5-14, Croatia, 2007.

[20] Frans Veldman, "Heuristic Anti-Virus Technology", International Virus Bulletin Conference, pp. 67-76, USA, 1993, available at `http://mirror.sweon.net/madchat/vxdevl/vdat/epheurs1.htm`.

[21] Jay Munro, "Antivirus Research and Detection Techniques", Antivirus Research and Detection Techniques, ExtremeTech, 2002, available at `http://www.extremetech.com/article2/0,2845,367051,00.asp`.

[22] D.W. Aha, D. Kibler, M.K. Albert, "Instance-based learning algorithms", Journal of Machine Learning, Vol. 6, pp. 37-66, 1991.

[23] M.E. Maron, J.L. Kuhns, "On relevance, probabilistic indexing and information retrieval", Journal of the Association of Computing Machinery, 7(3), pp.216-244, 1960.

[24] Y. Freund, R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting", Journal of Computer and System Sciences, No. 55, pp. 23-37, 1997

[25] J.R. Quinlan, "C4.5: Programs for machine learning", Morgan Kaufmann, USA, 1993.

[26] I.H. Witten, E. Frank, "Data mining: Practical machine learning tools and techniques", Morgan Kaufmann, 2nd edition, USA, 2005.

[27] VX Heavens Virus Collection, VX Heavens website, available at `http://vx.netlux.org`

[28] J. Oberheide, E. Cooke, F. Jahanian. "CloudAV: N-Version Antivirus in the Network Cloud", USENIX Security Symposium, pp. 91-106, USA, 2008.

[29] T. Fawcett, "ROC Graphs: Notes and Practical Considerations for Researchers", TR HPL-2003-4, HP Labs, USA, 2004.

[30] S.D. Walter, "The partial area under the summary ROC curve", Statistics in Medicine, 24(13), pp. 2025-2040, 2005.

[31] T.M. Cover, J.A. Thomas, "Elements of Information Theory", Wiley-Interscience, 1991.