# A Survey of Data Mining Techniques for Malware Detection using File Features

Muazzam Siddiqui
Institute of Simulation & Training
University of Central Florida
siddiqui@mail.ucf.edu

Morgan C. Wang
Department of Statistics and Actuarial Sciences
University of Central Florida
cwang@mail.ucf.edu

Joohan Lee
School of Electrical Engineering & Computer Science
University of Central Florida
jlee@cs.ucf.edu

## ABSTRACT

Malicious programs pose a serious threat to computer security. Traditional approaches using signatures to detect malicious programs pose little danger to new and unseen programs whose signatures are not available. The focus of the research is shifting from using signature patterns to more generalized schemes that do not assume any prior encounter with the malwares in focus. One such research area is using data mining techniques to detect novel malwares. This paper presents a survey of data mining techniques for malware detection using file features. While such surveys have already been conducted, the contribution made by this paper is, the categorization of these techniques by the data type used and since it is limited to the techniques using features gathered from the malware programs and/or emails only, it provides more depth in this area.

## Categories and Subject Descriptors

K.6.5 [**MANAGEMENT OF COMPUTING AND INFORMATION SYSTEMS**]: Security and Protection—*Invasive software*; H.2.8 [**DATABASE MANAGEMENT**]: Database Applications—*Data mining*; I.2.6 [**ARTIFICIAL INTELLIGENCE**]: Learning—*Concept learning, Connectionism and neural nets*

## General Terms

Survey, Security

## Keywords

survey, malware detection, data mining, machine learning, n-grams, instruction sequences, system calls

## 1. INTRODUCTION

Computer virus detection has evolved into malware detection since Cohen first formalized the term computer virus in 1983 [5]. Malwares can be classified into virus, worms, trojans, spywares, adwares and a variety of other classes and subclasses that sometimes overlap and blur the boundaries among these classes [17]. Both traditional signature based detection and generalized approaches can be used to identify these malicious programs. To avoid detection by the traditional signature based algorithms, a number of stealth techniques have been developed by the malicious code writers. The inability of traditional signature based detection approaches to catch these new breed of malicious programs has shifted the focus of malware research to find more generalized and scalable features that can identify malicious behavior as a process instead of a single signature action.

Data mining has been the focus of many virus researchers in the recent years to detect unknown viruses. A number of classifiers have been built and shown to have very high accuracy rates. Most of these classifiers use features extracted from executable programs by applying reverse engineering techniques. Besides binaries, email corpses, network traffic data and honeypot data are also mined for malicious activities.

While computer security is an umbrella term covering preventive measures like vulnerability analysis, as well as the detection of malwares, intruders, spam, phishing etc., this survey limits itself to the host based malware detection using data mining techniques on features extracted from files and/or emails. The following definitions describe the scope of this survey.

### 1.1 Malware Detection

The detection of malicious programs is termed as malware detection. This, differs from intrusion detection, which has a much wider scope as an intruder can be a human and/or a program. Consequently, intrusion detection can be carried out at a network scale using network traffic data or at a host level using data internal to the host. As stated previously, this survey focuses on the host based techniques using data extracted from programs.

### 1.2 File Features

Features extracted from programs are termed, in this paper, as file features. Various reverse engineering techniques at different stages are applied to extract byte sequences, instruction sequences, API call sequences etc.

### 1.3 Static Analysis Vs Dynamic Analysis

The analysis of programs to extract features can roughly

be divided into static and dynamic analysis. In the static analysis the code of the program is examined without actually running the program while in dynamic analysis the program is executed in a real or virtual environment.

## 1.4 Anomaly Detection Vs Misuse Detection

Anomaly detection systems look for deviations from a normal profile while the misuse detection is focused on specific malicious activities and behavior. In a data mining environment, if the input data source is only benign programs then the detection type will be anomaly detection while misuse detection results, if the features are extracted from malicious programs only. However in most of the detection systems, data is extracted from a collection of benign and malicious programs, hence, resulting in a hybrid detection system.

# 2. MALWARE DETECTION TECHNIQUES

The most common method of applying data mining techniques for malware detection start from generating a feature set. These features include hexadecimal byte sequences (later termed as n-grams in the paper), instruction sequences, API/system call sequences etc. This survey classifies the detection methods based upon the feature set they are operating on, the type of analysis to extract data from the programs (static/dynamic) and the type of detection strategy (Anomaly/Misuse).

## 2.1 N-grams

N-grams is a sequence of bytes of fixed or variable length, extracted form the hexadecimal dump of an executable program. Here, we are using n-grams as a general term for both overlapping and non-overlapping byte sequences. For the sake of this paper, we define n-grams to remain at a syntactic level of analysis.

### *2.1.1 Dynamic Misuse Detection*

The first major work that used data mining techniques for malware research was an automation of signature extraction for viruses by [8]. The viruses were executed in a secured environment to infect decoy programs. Candidate signatures of variable length were extracted by analyzing the infected regions in these programs that remained invariant from one program to another. Signatures with lowest estimated false positive probabilities were chosen as the best signatures. To handle large, rare sequences, a trigram approach was borrowed from speech recognition where the large sequence was broken down into trigrams. Then, a simple approximation formula can be used to estimate the probability of a long sequence by combining the measured frequencies of the shorter sequences from which it is composed. To measure algorithms effectiveness candidate signatures were generated and their estimated and actual probabilities were compared. Most of the signatures fell short of this criteria and were considered bad signatures. Another measure of effectiveness was false positive record of signatures, which was reported to be minimal. However no numerical values were provided for estimated and actual probability comparison and false positives.

### *2.1.2 Dynamic Hybrid Detection*

This was followed by the pioneering work of [6] where they extended the n-grams analysis to detect boot sector viruses using neural networks. The n-grams were selected based upon the frequencies of occurrence in viral and benign programs. Feature reduction was obtained by generating a 4-cover such that each virus in the dataset should have at least 4 of these frequent n-grams present in order for the n-grams to be included in the dataset. A validation set classification accuracy of 80-80% was obtained on viral boot sector while clean boot sectors received a 100% classification.

In the continuation of their work in [6], [4] used n-grams as features to build multiple neural network classifiers and adopted a voting strategy to predict the final outcome. Their dataset consisted of 53902 clean files and 72 variant sets of different viruses. For clean files, n-grams were extracted from the entire file while only those portions of a virus file are considered that remain constant through different variants of the same virus. A simple threshold pruning algorithm was used to reduce the number of n-grams to use as features. The results they reported are not very promising but it still presents a thorough work using neural networks.

### *2.1.3 Static Hybrid Detection*

The most recent work that brought the data mining techniques for malware detection in the limelight was done by [14]. They used three different types of features and a variety of classifiers to detect malicious programs. Their primary dataset contained 3265 malicious and 1001 clean programs. They applied RIPPER (a rule based system) to the DLL dataset. Strings data was used to fit a Naive Bayes classifier while n-grams were used to train a Multi-Naive Bayes classifier with a voting strategy. No n-gram reduction algorithm was reported to be used. Instead data set partitioning was used and 6 Naive-Bayes classifiers were trained on each partition of the data. They used different features to built different classifiers that do not pose a fair comparison among the classifiers. Naive-Bayes using strings gave the best accuracy in their model.

Extending the same ideas [13], created MEF, Malicious Email Filter, that integrated the scheme described in [14] into a Unix email server. A large dataset consisting of 3301 malicious and 1000 benign programs was used to train and test a Naive-Bayes classifier. N-grams were extracted by parsing the hexdump output. For feature reduction the dataset was partitioned into 16 subsets. Each subset is independently trained on a different classifier and a voting strategy was used to obtain the final outcome. The classifier achieved 97.7% detection rate on novel malwares and 99.8% on known ones. Together with [14], this paved the way for a plethora of research in malware detection using data mining approaches.

A similar approach was used by [9], where they built different classifiers including Instance-based Learner, TFIDF, Naive-Bayes, Support vector machines, Decision tree, boosted Naive-Bayes, SVMs and boosted decision tree. Their primary dataset consisted of 1971 clean and 1651 malicious programs. Information gain was used to choose top 500 n-grams as features. Best efficiency was reported using the boosted decision tree J48 algorithm.

[3] created class profiles of various lengths using different n-gram sizes. The class profile length was defined by the number of most frequent n-grams within the class. These frequent n-grams from both classes were combined to form, what they termed as relevant n-grams, for each profile length. Experiments were conducted on a set of 250 benign and 250 malicious programs. For classification, Dampster-Shafer

theory of Evidence was used to combine SVM, decision tree and IBK classifiers. They compared their work with [9] and reported better results.

In a related work, [2] used a Common N-Gram classification method to create malware profile signatures. Similar class profiles of various lengths using different n-gram sizes were created. The class profile length was defined by the number of most frequent n-grams with their normalized frequencies. A k-nearest neighbor algorithm was used to match a new profile instance with a set of already created signature profiles. They experimented with combinations of different profile lengths and different n-gram sizes. A five-fold cross validation method gave 91% classification accuracy.

Building upon their previous work in [21], [22] experimented with a larger data collection with 790 infected and 80 benign files, thus landing in our static hybrid detection category. N-grams were extracted from octal dump of the program, instead of usual hexdump and then converted to short integer values for SOM input. Using the SOM algorithms, VirusDetector was created that was used as a detection tool, instead of a visualization tool. VirusDetector achieved an 84% detection with a 30% false positive rate. the technique is able to cater polymorphic and encrypted viruses.

In a recent n-grams based static hybrid detection approach [7] used intra-family and inter-family support to select and reduce the number of features. First, the most frequent n-grams within each virus family were selected. Then the list was pruned to contain only those features that have a support threshold higher than a given value amongst these families. This was done for various n-gram sizes. Experiments were carried out on a set of 3000 programs, 1552 of which were viruses, belonging to 110 families, and 1448 were benign programs. With ID3, J4 decision tree, Naive Bayes and SMO classifiers, they compared their results with [14] and claimed better overall efficiency. In search of optimal feature selection criteria, they experimented with different n-gram sizes and various values of intra-family and inter-family selection thresholds. They reported better results with shorter sequences. For longer sequences, a low inter-family threshold gave better performance. Better performance was also noted, when features were in excess of 200.

#### 2.1.4 Static Misuse Detection

[21] presented a static misuse method to detect viruses using self organizing maps (SOM). They claimed that each virus has its own DNA like character that changes the SOM projection of the program that it infects. N-grams were extracted from the infected programs and SOMs were trained on this data. Since the method only looks for change in the SOM projection as a result of virus infection, it is able to detect polymorphic and metamorphic malwares, even in the encrypted state. Experiments were performed on a small set of 14 viral samples. The algorithm was successfully able to detect virus patterns in the infected files.

### 2.2 API/System calls

An application programming interface (API) is a source code interface that an operating system or library provides to support requests for services to be made of it by computer programs [1]. For operating system requests, API is interchangeably termed as system calls. An API call sequences captures the activity of a program and, hence, is an excellent candidate for mining of any malicious behavior.

#### 2.2.1 Dynamic Anomaly Detection

[10] added a temporal element to the system call frequency and calculated the frequency of system call sequences within a specific time window. These windows were generated for independent peers and a correlation among them indicated a fast spreading worm. Similarity measures were calculated using edit distance on ordered sets of system calls windows and intersection on unordered sets. These similarity measures gave the probabilities of two peers running the same worm invoking the same system call sequence in a given time window. The technique works well for polymorphic worms.

#### 2.2.2 Static Hybrid Detection

In the malware detection realm, the most important work using API sequences was done by [16]. They created a signature based detection system called Static Analyzer of Vicious Executables (SAVE) that compares API sequences extracted from programs to sequences from a signature database. For matching cosine measure, extended Jaccard measure and Pearson correlation measures were used. Final outcome was the mean of all three measures. Using these statistical measures for similarity enables SAVE to capture polymorphic and metamorphic malwares, for which, traditional signature detection system are deemed incapable of.

In a recent approach using API execution sequences, [20] developed Intelligent Malware Detection System (IMDS). IMDS used Objective-Oriented Association (OOA) mining based classification and is composed of a PE parser, an OOA rule generator and a rule based classifier. For rule generation, they developed a OOA Fast FP-Growth algorithm, an extension of FP-Growth algorithm, and claimed better performance than the Apriori algorithm for association rule generation. The experiments were conducted on a large data set consisted of 17366 malicious and 12214 benign executables. For detection, a Classification Based on Association rules (CBA) technique was used. They compared their results with popular anti-virus programs like Norton, McAfee, Dr Webb and Kaspersky and claimed better efficiency on polymorphic and unknown virus detection. The IMDS was able to detect all the polymorphic viruses and 92% of the unknown viruses. For polymorphic virus detection a different dataset was created where for each virus a set of polymorphic versions were generated. IMDS was also compared with Naive-Bayes, J48 decision tree and SVM classifiers and proved to churn out better detection and false positive rates. A smaller dataset was sampled from the collection for this comparison.

### 2.3 Assembly Instructions

In more recent developments, it was emphasized that since n-grams fail to capture the semantics of the program, instruction sequences should be used instead. Unlike extracting n-grams from hexdumps, instructions need disassembling the binaries. We define, assembly instruction features to be instructions of variable and/or fixed lengths extracted from disassembled files.

The first major work to include instructions in its features was Portable Executable Analysis Tool (PEAT) [19]. PEAT used a number of other features also, so it will be explained in detail in the Hybrid Features section. Similarly [12] used instruction sequences along with other features and will be

discussed in the Hybrid Features section.

### 2.3.1 Static Misuse Detection

[11] created a malware phylogeny using permutation of code. They extracted instruction sequences of fixed length (termed as n-grams in their paper) and created two datasets. First dataset contained all the n-grams while the second dataset featured n-perms, all possible permutations of the n-grams in the first dataset. Similarity scores were calculated using a combination of TFxIDF weighting and cosine similarity measures. Clustering on these similarity scores provided the phylogeny model. To measure the effectiveness of each feature set, an experiment was conducted on a small data collection of 170 samples. The results showed that n-perms produce higher similarity scores for permuted programs and produce comparable phylogeny models. A study was conducted to explore how the generated malware phylogeny can be used in naming and classification of malwares. Using a small dataset of 15 samples including an unknown malware, the created phylogeny model successfully identified the unknown malware family. The phylogeny model was also able to identify naming inconsistencies for names assigned by commercial antiviruses.

### 2.3.2 Static Hybrid Detection

Another recent work that solely used variable length instruction sequences, was [15]. After disassembly, instruction sequences were selected based upon the frequency of occurrence in the entire dataset. A Chi-Square test was performed for feature reduction. They built logistic regression, neural network and decision tree models and reported 98.4% detection rate with decision tree model.

## 2.4 Hybrid Features

A number of techniques used a collection of features, either for different classifiers, or as a hybrid feature set for the same classifier. The work done by [14] used three different features including n-grams, DLL usage information and strings extracted from the binaries. Since this work was made famous for its usage of n-grams, we finally decided to place it in the n-grams section.

### 2.4.1 Static Anomaly Detection

[19] developed PEAT (The Portable Executable Analysis Tool) to detect structural anomalies inside a program. PEAT rested on the basic principle that the inserted code in a program disrupts its structural integrity and hence by using statistical attributes and visualization tools this can be detected. The visualization tools plot the probability of finding some specific subject of interest in a particular area of the program. These subjects include sequence of bytes, their ASCII representation, their disassembly representation and memory access via register offsets. Statistical analysis was done on instruction frequencies, instruction patterns, register offsets, jump and call offsets, entropy of opcode values and code and ASCII probabilities. The experimental results were provided for only one malicious program.

### 2.4.2 Static Hybrid Detection

[12] created a hybrid feature set using n-grams, instruction sequences and API calls. For feature selection, Information Gain was used. They experimented with two non-disjoint datasets. The first dataset contains a collection of 1,435 executables, 597 of which are benign and 838 are malicious. The second dataset contains 2,452 executables, having 1,370 benign and 1,082 malicious executables. In addition to the hybrid feature set (HFS), two other datasets were also created using n-grams (BFS) and assembly features (AFS). The accuracy of each of the feature sets was tested by applying a three-fold cross validation using classifiers such as SVM, decision tree, Naive Bayes, Bayes Net and Boosted decision tree. Experiments were performed for different n-gram sizes. For the first dataset, best efficiency was reported for HFS using an n-gram size of 6, resulting in 97.4% classification accuracy. HFS performed better for the second dataset too. To compare classification accuracies of the HFS and BFS, a pairwise two-tailed t-test was performed. The test statistically proved that HFS performed slightly better than BFS.

**Table 1: Summary**

| Features | Analysis Type | Detection Strategy | Examples |
|---|---|---|---|
| N-grams | Static | Hybrid | [7], [14], [2], [9], [13], [3], [22] |
| | | Misuse | [21] |
| | Dynamic | Hybrid | [4], [6] |
| | | Misuse | [8] |
| Instructions | Static | Hybrid | [15] |
| | | Misuse | [11] |
| Hybrid Features | Static | Anomaly | [19] |
| | | Hybrid | [12] |
| | Hybrid | Hybrid | [18] |
| API/System Calls | Static | Hybrid | [20], [16] |
| | Dynamic | Anomaly | [10] |

### 2.4.3 Hybrid Hybrid Detection

[18] presented a surveillance spyware detection system that used APIs and DLL usage information and changes in registry, system files/folders and network states to detect spywares. They collected 1147 samples over a period of time to experiment with. Information Gain and SVM runs were used for feature selection. An SVM was used to classify spywares from benign programs. They reported 97.9% detection with a 0.68% false positive rate.

# 3. CONCLUSIONS

In this paper we presented a survey of data mining approaches for malware detection using file features. We provided a classification hierarchy based upon the type of feature used, the method of program analysis (static/dynamic) and the type of detection used (anomaly/misuse).

Table 1 gives a summary of the research that we have surveyed in this paper.

# 4. REFERENCES


[1] Wikipedia API article. http://en.wikipedia.org/wiki/api.

[2] T. Abou-Assaleh, N. Cercone, V. Keselj, and R. Sweidan. Detection of new malicious code using n-grams signatures. In *Proceedings of Second Annual Conference on Privacy, Security and Trust*, pages 193–196, 2004.

[3] T. Abou-Assaleh, N. Cercone, V. Keselj, and R. Sweidan. N-gram-based detection of new malicious code. In *Proceedings of the 28th Annual International Computer Software and Applications Conference - Workshops and Fast Abstracts - (COMPSAC'04) - Volume 02*, pages 41–42, 2004.

[4] W. Arnold and G. Tesauro. Automatically generated win32 heuristic virus detection. In *Virus Bulletin Conference*, pages 123–132, 2000.

[5] F. Cohen. *Computer Viruses*. PhD thesis, University of Southern California, 1985.

[6] J. O. K. Gerald J. Tesauro and G. B. Sorkin. Neural network for computer virus recognition. *IEEE Expert*, 11(4):5–6, 1996.

[7] O. Henchiri and N. Japkowicz. A feature selection and evaluation scheme for computer virus detection. *icdm*, 0:891–895, 2006.

[8] J. O. Kephart and B. Arnold. Automatic extraction of computer virus signatures. In *Proceedings of the 4th Virus Bulletin Internation Conference*, pages 178–184, 1994.

[9] J. Z. Kolter and M. A. Maloof. Learning to detect malicious executables in the wild. In *Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.

[10] D. J. Malan and M. D. Smith. Host-based detection of worms through peer-to-peer cooperation. In *WORM '05: Proceedings of the 2005 ACM workshop on Rapid malcode*, pages 72–80. ACM, 2005.

[11] A. L. Md. Enamul Karim, Andrew Walenstein and L. Parida. Malware phylogeny generation using permutations of code. *Journal in Computer Virology*, 1(1,2):13–23, 2005.

[12] L. K. Mohammad M. Masud and B. Thuraisingham. A scalable multi-level feature extraction technique to detect malicious executables. *Information Systems Frontiers*, 2007.

[13] M. G. Schultz, E. Eskin, E. Zadok, M. Bhattacharyya, and S. J. Stolfo. MEF: Malicious email filter: A UNIX mail filter that detects malicious windows executables. pages 245–252, 2001.

[14] M. G. Schultz, E. Eskin, E. Zadok, and S. J. Stolfo. Data mining methods for detection of new malicious executables. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 38–49, 2001.

[15] M. Siddiqui, M. C. Wang, and J. Lee. Data mining methods for malware detection using instruction sequences. Accepted at IASTED AIA 2008, 2008.

[16] A. H. Sung, J. Xu, P. Chavez, and S. Mukkamala. Static analyzer of vicious executables. In *20th Annual Computer Security Applications Conference*, pages 326–334, 2004.

[17] P. Szor. *The Art of Computer Virus Research and Defense*. Addison Wesley for Symantec Press, New Jersey, 2005.

[18] M.-Y. S. C.-H. W. P.-C. W. Tzu-Yen Wang, Shi-Jinn Horng and W.-Z. Su. A surveillance spyware detection system based on data mining methods. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 3236– 3241. IEEE, 2006.

[19] M. Weber, M. Schmid, M. Schatz, and D. Geyer. A toolkit for detecting and analyzing malicious software. In *Proceedings of the 18th Annual Computer Security Applications Conference*, page 423, 2002.

[20] Y. Ye, D. Wang, T. Li, and D. Ye. Imds: intelligent malware detection system. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1043–1047. ACM, 2007.

[21] I. Yoo. Visualizing windows executable viruses using self-organizing maps. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 82–89, 2004.

[22] I. Yoo and U. Ultes-Nitsche. Non-signature based virus detection: Towards establishing unknown virus detection technique using som. *Journal in Computer Virology*, 2(3):163–186, 2006.