VIRUS DETECTION USING DATA MINING TECHINQUES[i]

Jau-Hwang WANG, Peter S. DENG,
Yi-Shen FAN, Li-Jing JAW,Yu-Ching LIU
Department of Information Management
Central Police University
Tao-Yuan, Taiwan, ROC 333

ABSTRACT

Malicious executables are computer programs, which may cause damages or inconveniences for computer users when they are executed. Virus is one of the major kinds of malicious programs, which attach themselves to others and usually get executed before the host programs. They can be easily planted into computer systems by hackers, or simply down loaded and executed by naïve users while they are browsing the web or reading E-mails. They often damage its host computer system, such as destroying data and spoiling system software when they are executed. Thus, to detect computer viruses before they get executed is a very important issue. Current detection methods are mainly based on pattern scanning algorithms. However, they are unable to detect unknown viruses. In this paper, an automatic heuristic method to detect unknown computer virus based on data mining techniques, namely *Decision Tree* and *Naïve Bayesian network* algorithms, is proposed and experiments are carried to evaluate the effectiveness the proposed approach.

Keywords: *Computer Security, Virus Detection, Data Mining, Decision Tree. Naïve Bayesian Network.*

1、INTRODUCTION

Malicious executables [1] are computer programs, which may cause damages or inconveniences for computer users when they are executed by computers. Malicious codes are very easy to bypass the system security measures and get planted into computer systems. The Internet was designed and implemented with the intension to be used in rather limited environment, such as academic and research institutes. The security issue was often of secondary importance. Even after 1990s, when the networks are widely used by general public, the Internet is still very vulnerable due to the diversity of network applications. Consequently, malicious codes can be easily planted into computer systems by hackers, or simply down loaded by naïve users from the Internet while browsing the web. Since the execution of malicious codes can cause damages, the detection of malicious codes in computer systems has become a very important issue. Currently, the techniques for malicious program detection are mainly based on heuristic analysis and pattern scanning. The heuristic analysis procedures are very expensive and time consuming. Data mining has been loosely defined as the process of extracting interesting patterns from a large amount of database records. As the data mining technology becomes more and more matured, it is feasible to develop automatic program analysis and classification techniques for malicious program detection.

Computer virus [2] is one of the major kinds of malicious programs. Computer viruses are small computer programs, which attach themselves to others and usually get executed before the host programs. Commonly, they operate in two phases, namely the replicate phase and the active phase. In the replicate phase the viruses reproduce and attach to other programs but cause no immediate ill effects on the host systems. Usually they remain benign until some trigger conditions--such as the passage of a given time or the occurrence of a specific date--occur, at which the viruses change to the active phase and often damaging its host computer system, such as destroying data and spoiling

0-7803-7882-2/03/$17.00©2003 IEEE

system software. Most viruses also include a string of characters acts as a marker to indicate that a program has been infected. When a virus replicates, it selects an executable file and checks to see if the file has been infected. If the virus finds that the selected file is uninfected, it inserts a copy of itself into that file, otherwise another program is selected.

Anti-virus scanning is one of the major countermeasures for computer virus. Virus scanners are computer programs that examine each piece of computer software for the occurrences of virus patterns. Although virus scanners are good against known virus and other specific patterns, they won't work against viruses with new "breads". In order to remain effective, the virus patterns of a scanner must be updated very often. However, if new virus appears at a rate of several per day[1], keeping the virus patterns up-to-date may not be practical. Thus, it is of highly demand to develop a generic scanner, which is able to detect new viruses without need to update its pattern database. Although it is proven [2] that the problem of distinguishing a virus from a non-virus program is unsolvable in general case, some generic detection is still possible [4]. It is based on analyzing a program for a vector of features typical or not typical for viruses. The process of feature discovery and possibly together with a set of rules is known as heuristic approach. Recently more and more researchers are looking towards heuristic approach as at least a partial solution to the problem [4,5,6]. This paper proposed a detection method for unknown computer virus using data mining algorithms. Feature vectors were automatically extracted to capture the characteristics of virus programs. The feature vectors were then used to train classification models for virus detection. Experiments were carried out to measure the detection rates and accuracies of the classification algorithms proposed.

[1] Currently, the number of new viruses created is around 800 to 1000 a month [3].

2、RELATED WORK

Virus detection methods mainly can be divided into two categories, namely static analysis and dynamic analysis [7].

2.1、Static Analysis

With static analysis, a virus is detected by analyzing the virus codes, the infected files or records. Methods in this category include virus scanners, cryptographic checksums, integrity shells, and printable string examiners [2,6]. A "Virus Scanner" is a program that examines system files or records for the occurrences of virus patterns. Scanners are often used as a bootstrap check for known viruses. However, they have some major problems. Firstly, they are only good against known viruses and not very good against evolutionary or new "breads" viruses. Secondary, they tend to take a noticeable amount of time to scan a system or networks for the patterns. Thirdly, a scanner or its virus pattern database must be updated very often to remain effective.

The integrity shell and cryptographic checksum [2,8] are more like virus defense methods than virus detection methods, and both use redundancy to detect changes caused by viruses. An integrity shell is a form of automated fault tolerance and change control, and often with automatic correction capability. An integrity shell only interprets a program or command if only that everything the program or command depends on is unchanged, otherwise the program or command may be replaced or forgot according the specification. A cryptographic checksum is like a fingerprint. The files in a system are encrypted using a secret key and then a checksum for each of the encrypted file is created. If a file is subsequently modified, its fingerprint changes as well. Since it is very unlikely for the virus to correctly modify a file and its checksum without knowing the encryption key, the changes made by virus modification will be detected using the checksum technique.

Often strings contained in a binary can be used to distinguish malicious executables from clean benign programs [6]. Those strings mostly consist of reused code fragments, file names, author signatures, system resource information, etc, and have been used by the anti-malicious executable community as signatures for malicious programs.

2.2 、 Dynamic Analysis

With dynamic analysis, viruses are detected as they are executed. Methods in this category include function or system call tracers, machine emulators, logic analyzers, and network sniffers. Function or system call tracers employ intrusion detection techniques. "Normal" and "abnormal" system or function sequences are supplied to train a classification model, which can then be used to distinguish abnormal sequence from a normal one [9]. A machine emulator provides a virtual environment for program execution. As the suspect virus codes are executed, the after-effects of the execution are examined to determine if the program executed is a virus. Logic analyzer intercepts every instruction of the traced process to gather the process's information. Besides that it may slow down the execution by many magnitude, it may also produce too much information.

2.3 、 Discussion

Dynamic analysis has the advantage of being fast and accurate. However, it is difficult to traverse all possible paths through a program code. Most recent researches on virus detection used static analysis [5,6,8].

Kephart, et al, [5] from IBM have developed an automatic method to extract signatures from virus codes. The extraction method is based on the exhausted computation to find the byte sequences which are statistically most likely to appear in virus or infected codes and least likely to appear in non-infected codes. The signatures can then used to scan programs for virus infection. Lai, et al [8] derived common code words from virus codes and used them to match program files for virus infection. Schultz, et al [6] derived virus features by combining system resources used by the virus (such as dynamically linked library routine calls), strings extracted by GNU strings command, and byte sequences which were only found in malicious executables. The features were then used to train classifiers for unknown virus detection.

3 、 PROPOSE APPROACH

Program analysis can be conducted at least in three different levels, binary or bit sequence level, machine instruction or assembly language level, and high-level language levels. Bit sequence analysis has the advantage to be able to provide the most detail and exact content of the programs. However, the information represented in bit sequence level may provide too much detail information and not suitable for pattern extraction. This is similar to the fact that although all text documents are encoded as ASSCII codes, which however were seldom used as the semantic units for analysis in information retrieval [10] community. Due to the development of reverse engineering techniques [11], such as de-assembler and de-compiler, in recent years it is possible to analyze program in assembly language or even in high-level language levels. We believe that analyzing malicious programs in higher level will be able to extract more meaningful patterns for classification. This research proposed to detect unknown virus codes using data mining techniques. The steps of our experiment is as shown in Figure 1:

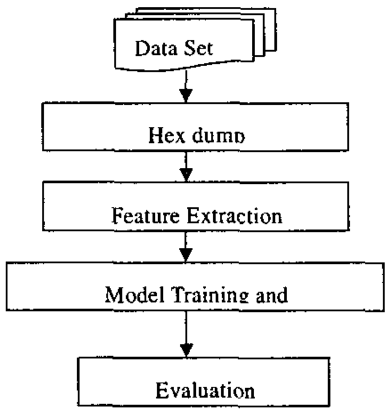

Figure 1. The Experiment Steps

3.1 、 Data Sets and Transformation

We downloaded a collection of virus codes at http://www.cs.columbia.edu/ids.mef/software/. The data set consists of 3265 malicious binaries and 1001 benign programs A subset of it, which consists of 125 benign programs and 875 malicious codes of file infectors, was dumped to text form. Each program was then transformed to a vector of the form $X=(X_1,X_2,\ldots,X_n)$, where each element is a byte sequence of each instruction of the program. Two data sets were created, namely Data Set One, and Data Set Two. Data Set One consists of the 125 benign and 875 virus programs. Data Set Two, which consists of 345 of almost equal number of codes in both benign and virus classes, was randomly selected from Data Set One.

3.2、Feature Selection and Extraction

The distinguishing power of each feature is derived by computing its *information gain* based on its frequencies of appearances in the virus class and benign class. For example, Data Set One consists of 125 benign programs and 875 virus codes. The expected information, E(X), needed to classify the data set is calculated by the following equation [12]：

$$-\left(\frac{|benign|}{|X|}\log_2\frac{|benign|}{|X|}+\frac{|virus|}{|X|}\log_2\frac{|virus|}{|X|}\right)$$

$$=\frac{-125}{875+125}\log_2\left(\frac{125}{875+125}\right)+\frac{-875}{875+125}\log_2\left(\frac{875}{875+125}\right)$$

If the data set is further partitioned by feature X_i, the information gain $I(X, X_i)$ is $E(X) - E(X|X_i)$, where $E(X|X_i)$ is equal to: Probability(X is virus | $X_i = 0$) × Probability(X is benign | $X_i = 0$) + Probability(X is virus | $X_i = 1$) × Probability(X is benign | $X_i = 1$). The information gains for each feature (which consisting of the first two bytes of each instruction) are shown in Table 1. For example, the feature "0128", I(X,"0128") = E(X) − E(X|"0128") =

$$E(X)-\frac{124+872}{1000}\times\left(\frac{-124}{124+872}\log_2\left(\frac{124}{124+872}\right)+\frac{-872}{124+872}\log_2\left(\frac{872}{124+872}\right)\right)$$
$$-\frac{1+3}{1000}\times\left(\frac{-1}{1+3}\log_2\left(\frac{1}{1+3}\right)+\frac{-3}{1+3}\log_2\left(\frac{3}{1+3}\right)\right)=0.00033448$$

Table 1. The Information Gains for Each Feature

Features	Number of Appearance in Benign Class		Number of Appearance In Virus Class		Information Gain
	0	1	0	1	
00D7	125	0	874	1	0.000192748
260B	125	0	873	2	0.000385703
C14D	125	0	872	3	0.000578865
6A6B	125	0	871	4	0.000772234
D382	125	0	870	5	0.000965811
FF6D	124	1	875	0	0.003005065
009D	124	1	874	1	0.001196369
FFF2	124	1	873	2	0.00063299
0128	124	1	872	3	0.00033448
00A5	123	2	873	2	0.002400241
···	···	···	···	···	···
Total		2		8	M(I)=0.002

Features with negligible information gains can then be removed to reduce the number of features and speed up the classification process.

(3)、Model Training and Classification

Each data set is further partitioned at ratio 7:2:1 into training set, testing set, and validating set. Each data set was fed to the Naïve Bayesian and Decision tree classifiers of Insightful Miner to conduct the experiments. The experiment is repeated five times using random sub-sampling holdout method [12] and the detection rates and accuracies obtained from the five iterations are averaged to obtained the experiment results.

(4)、Notations and Evaluation Measures

The following measures [12] are used to evaluate the correctness of the classification models for unknown virus detection:

A. **True Positive (TP)**：Number of programs correctly classified as virus codes.

B. **True Negative (TN)**：Number of programs correctly classified as benign programs.

C. **False Positive (FP)** : Number of benign programs incorrectly classified as virus codes.

D. **False Negative (FN)** : Number of virus codes misclassified as benign programs.

E. **Detection Rate (DR)** = $\frac{TP}{TP+FN}$.

F. **False Positive Rate (FPR)** = $\frac{FP}{TN+FP}$.

G. **Accuracy (ACY)** = $\frac{TP+TN}{TP+TN+FP+FN}$.

4、EXPERIMENT RESULTS

(1)、Experiment Results from Data Set One

Two experiments were conducted on this data set. Firstly, the first byte (mainly the op-code) of each instruction was used as an element for the feature vector. Secondary, both the first two bytes (mainly the op-code and the first operand) of each instruction were taken as an element. The experiment results are shown in Table 2. The results show that the unknown virus detection rates for the Decision Tree and Naïve Bayesian classifiers are 93.2% and 76.1%, with accuracies of 89.5% and 73.3% respectively, when each feature element consists of only the op-code of each instruction. The unknown virus detection rates for the Decision Tree and Naïve Bayesian classifiers raise to 94.3% and 80.7%, and accuracies also raise to 91.4% and 77.1%, when the op-code and the first operand are used as a feature element. We may conclude that when each feature contains more information, the classifiers perform better. We also observed that the Decision Tree algorithm outperforms the Naïve Bayesian network algorithm both in detection rate, false positive rate, and accuracy.

Table 2. Experiment Results from the Data Set One

Algorithms	Feature Element length	TP	TN	FP	FN	DR (%)	FPR (%)	ACY (%)
Naïve Bayesian	1 Byte	67	10	7	21	76.1	41.2	73.3
	2 Bytes	71	10	7	17	80.7	41.2	77.1
Decision Tree	1 Byte	82	12	5	6	93.2	29.4	89.5
	2 Bytes	83	13	4	5	94.3	23.5	91.4

(2)、Experiment Results from Data Set Two

We observed that when data set consists of almost equally mix of benign and virus programs, the detection rates and accuracies of both classifiers dropped significantly. In certain case the accuracy even dropped up to 25%. The false positive rates also dropped to some extent. The cause of this phenomenon yet needs to be further studied. However, we can conclude that it is important to have a good mix of virus and benign programs to train the classification models, since it significantly affects the effectiveness of both classifiers.

Table 3: Experiment Results from Data Set Two

Algorithms	Feature Element	TP	TN	FP	FN	DR (%)	FPR (%)	ACY (%)
Naïve Bayesian	1 Byte	15	11	6	3	83.3	35.2	74.3
	2 Bytes	15	12	5	3	83.3	29.4	77.1
Decision Tree	1 Byte	13	14	3	5	72.2	12.5	79.4
	2 Bytes	11	12	5	7	61.1	29.4	65.7

5、CONCLUSIONS

Anti-virus scanning is one of the major countermeasures for computer virus detection. However, they mainly rely on human experts to extract the virus patterns and are only good for detecting known viruses, and often not suitable for detecting evolutional and unknown viruses. This research proposed a data mining approach to automatically extract virus features from virus programs and used the features to train classification model for unknown virus detection. Experiments were conducted to evaluate the proposed method. The experiment results show that the detection rates for the Decision Tree and Naïve Bayesian classifiers are 94.3% and 80.7%, and the accuracies are 91.4% and 77.1%, respectively, which are very promising. In general, the Decision Tree algorithm outperforms the Naïve Bayesian network classifier. Furthermore, when each feature element contains more information, both the classifiers perform better. It is also found that a good mix of virus and benign data set to train the classification models has significant effect on the effectiveness of both classifiers. However, to find a good mix is still opened for future work.

REFERENCES

[1] Stephen Cass, "Anatomy of Malice", *IEEE SPECTRUM*, November 2001.

[2] Fred Cohen, *A Short Course on Computer Viruses*, 2nd edition, John Wiley & Sons, Inc. 1994.

[3] George Lawton, "Virus Wars : Fewer Attacks, New Threats", *IEEE Computer*, Vol. 35, No. 12, 2002.

[4].Dmitry O. Gryaznov, "Scanners of The Year 2000: Heuristics", http://vx.netlux.org/texts/html/scan2000.html

[5] Jeffrey O. Kephart and William C. Arnold, "Automatic Extraction of Computer Virus Signatures", *Proceedings of the 4th Virus Bulletin International Conference*, 1994.

[6] Matthew G. Schultz, Eleazar Eskin, Erez Zadok and Salvatore J. Stolfo, "Data Mining Methods for Detection of New Malicious Executables", *The 2001 IEEE Symposium on Security and Privacy, Oakland,* CA, May 2001.

[7] Wietse Venema, "Strangers in the Night", Dr. Dobb's Journal, November, 2000, http://www.ddj.com/documents/s=879/ddj0011g/0011g.htm

[8], Zone-Chang Lai and Bing-Shin Tsai, "Codeword-based Virus Detection Using Virus Features," *Proceedings of 2001 IPPR Conf. on Computer Version, Graphics and Image Processing.* Aug. 2001, Ping-Ton, Taiwan.

[9] Wenke Lee and Salvatore J. Stolfo, "Data Mining Approaches for Intrusion Detection", http://www.cs.columbia.edu/~sal/hpapers/USENIX/usenix.html.

[10] Gerard Salton, *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*, Addison Wesley, 1989.

[11] Cristina Cifuentes, *Reverse Compilation Techniques,* PhD. Dissertation, Queensland University of Technology, Department of Computer Science, 1994.

[12] Jiawei Han and Micheline Kamber, *Data Mining: Concepts and Techniques,* Morgan Kaufmann Publishers, 2001.

[13] Roger A. Grimes, *Malicious Mobile Code – Virus Protection for windows*, 1st edition, O'Reilly & Associates, Inc., 2001.

---

[i] This research is partially supported by NSC, ROC, under grand no. 92-2213-E-015-006