# New Malicious Code Detection Based on N-Gram Analysis and Rough Set Theory

**Conference Paper** · November 2006

DOI: 10.1007/978-3-540-74377-4_65 · Source: DBLP



**5 authors**, including:

Boyun Zhang
Hunan police academy
**21** PUBLICATIONS **80** CITATIONS
SEE PROFILE

Jianping Yin
National University of Defense Technology
**150** PUBLICATIONS **846** CITATIONS
SEE PROFILE

Jingbo Hao
National University of Defense Technology
**11** PUBLICATIONS **41** CITATIONS
SEE PROFILE

Shulin Wang
University of Kansas
**41** PUBLICATIONS **311** CITATIONS
SEE PROFILE

# New Malicious Code Detection Based on N-Gram Analysis and Rough Set Theory

Boyun Zhang[1,2], Jianping Yin[1], Jingbo Hao[1], Shulin Wang[1], and Dingxing Zhang[1]

[1] School of Computer Science, National University of Defense Technolgy, Changsha 410073, China
{hnjxzby,dingxingzhang}@yahoo.com.cn, {jpyin,hjb}@nudt.edu.cn, jt_slwang@hnu.cn
[2] Department of Computer Science, Hunan Public Security College, Changsha 410138, China

**Abstract.** Motivated by the standard signature-based technique for detecting viruses, we explore the idea of automatically detecting malicious code using the N-gram analysis. The method is based on statistical learning and not strictly dependent on certain viruses. We propose the use of rough set theory to reduce the feature dimension. An efficient implementation to calculate relative core, based on positive region definition is presented also. The k nearest neighbor and support vector machine classifiers are used to categorize a program as either normal or abnormal. The experimental results are promising and show that the proposed scheme results in low rate of false positive.

## 1 Introduction

Since the appearance of the first computer virus in 1986, a significant number of new viruses have appeared every year. This number is growing and it threatens to outpace the manual effort by anti-virus experts in designing solutions for detecting them and removing them from the system [1]. Excellent technology exists for detecting known viruses. Programs such as Norton Anti-Virus are ubiquitous. These programs search executable code for known patterns. One shortcoming of this method is that we must obtain a copy of a malicious program before extracting the pattern necessary for its detection.

And then there have been few attempts to use machine learning and data mining for the purpose of identifying new or unknown malicious code [2-4]. In addition there are other methods of guarding against malicious code, such as object reconciliation, which involves comparing current files and directories to past copies. One can also compare cryptographic hashes. These approaches are not based on data mining.

In this paper, we explore solutions based on machine learning and not strictly dependent on certain viruses. It would not be feasible to design a general anti-virus tool that could replace a human expert or be as reliable as the exact solutions for known viruses, but such a solution would be of a great benefit in warning against new viruses, in aiding experts in finding a good signature for a new virus, and in adaptable solutions for different users.

Y. Wang, Y. Cheung, and H. Liu (Eds.): CIS 2006, LNAI 4456, pp. 626–633, 2007.

In the following sections, we first detail the feature selection based on N-gram and Information Gain, and then a new feature reduction method by using rough set theory is proposed. Section 3 shows the experiment results. We state our plans for future work in Section 4.

## 2 Detection Engine

### 2.1 Feature Extraction

The idea of using n-grams for malicious code analysis is not new [1, 5], but we did not find many reported results. Before extracting features of each file, we first introduce n-gram analysis. An n-gram [6] is a subsequence of $N$ consecutive tokens in a stream of tokens. N-gram analysis has been applied in many tasks, and is well understood and efficient to implement. By converting a string of data to n-grams, it can be embedded in a vector space to efficiently compare two or more streams of data. Alternatively, we may compare the distributions of n-grams contained in a set of data to determine how consistent some new data may be with the set of data in question. An n-gram distribution is computed by sliding a fixed size window through the set of data and counting the number of occurrences of each "gram".

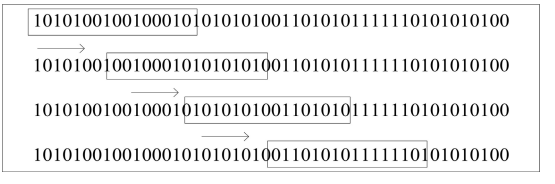


**Fig. 1.** Sliding window (window size=2 Byte)

**Table 1.** Feature Frequency Matrix. ($n$=2)

| Samples | Features | | | | | |
|---|---|---|---|---|---|---|
| | 0080 | 61C3 | 638D | 9090 | E020 | … |
| Win32.Alcaul.a | 2 | 4 | 21 | 8 | 79 | |
| Win32.Alcaul.b | 45 | 6 | 40 | 7 | 20 | |
| Win32.Alcaul.c | 11 | 11 | 18 | 8 | 14 | |
| Win32.Alcaul.e | 7 | 0 | 12 | 6 | 25 | |
| Win32.Alcaul.f | 9 | 20 | 15 | 5 | 27 | … |
| Win32.Alcaul.g | 11 | 0 | 17 | 3 | 20 | |
| Win32.Alcaul.h | 19 | 17 | 48 | 4 | 29 | |
| … | … | … | … | … | … | |

Figure 1 displays an example of a 2-byte window sliding right one byte at a time to generate each 2-gram. Each 2-gram is displayed in the highlighted "window". The choice of the window size depends on the application. First, the computational complexity increases exponentially as the window size increases. Data is considered a stream of tokens drawn from some alphabet. If the number of distinct tokens (or the size of the alphabet) is $X$, then the space of grams grows as $X^N$. In this work, we focus initially on 2-gram analysis of binary values of PE format file. After the preliminary

processing, the frequency matrix of data set is obtained. An example of frequency matrix is shown in table1 of our experiment.

### 2.2 Feature Selection Based on IG

For feature selection in our approach we adopt correlation measures based on the information theoretical concept of entropy, a measure of the uncertainty of a random variable. The distinguishing power of each feature is derived by computing its information gain (IG) based on its frequencies of appearances in the malicious class and benign class. Features with negligible information gains can then be removed to reduce the number of features and speed the classification process.

The entropy of a variable X is defined as:

$$H(X) = -\sum_i P(x_i)\log_2(P(x_i)), \tag{1}$$

And the entropy of $X$ after observing values of another variable $Y$ is defined as:

$$H(X \mid Y) = -\sum_j P(y_j)\sum_i P(x_i \mid y_j)\log_2(P(x_i \mid y_j)), \tag{2}$$

where $P(x_i)$ are the prior probabilities for all values of $X$, and $P(x_i \mid y_j)$ is the posterior probabilities of $x_i$ given the values of $y_j$. The amount by which the entropy of $X$ decreases reflects additional information about $X$ provided by $Y$ is called information gain, given by

$$IG(X \mid Y) = H(X) - H(X \mid Y). \tag{3}$$

Information gain tends to favor variables with more values and can be normalized by their corresponding entropy.

For our problem, the expected entropy calculated as:

$$\begin{aligned} H(X) = -[&P(x \text{ is } normal)\cdot\log_2 P(x \text{ is } normal) \\ &+ P(x \text{ is } abnormal)\cdot\log_2 P(x \text{ is } abnormal)]. \end{aligned} \tag{4}$$

If the data set is further partitioned by feature $y_i$, the conditional entropy $H(X \mid y_i)$ is calculated as:

$$\begin{aligned} H(X \mid y_i) = &-P(y_i = 0)\cdot[P(x \text{ is } normal \mid y_i = 0)\cdot\log_2 P(x \text{ is } normal \mid y_i = 0) \\ &+ P(x \text{ is } abnormal \mid y_i = 0)\cdot\log_2 P(x \text{ is } abnormal \mid y_i = 0)\ ] \\ &- P(y_i = 1)\cdot[\ P(x \text{ is } normal \mid y_i = 1)\cdot\log_2 P(x \text{ is } normal \mid y_i = 1) \\ &+ P(x \text{ is } abnormal \mid y_i = 1)\cdot\log_2 P(x \text{ is } abnormal \mid y_i = 1)\ ]. \end{aligned} \tag{5}$$

where $y_i = 0$ denotes that the feature $y_i$ do not appear in the samples, $y_i = 1$ denotes that the feature $y_i$ appears in the samples.

The information gains for each feature are detailed in table 2.

**Table 2.** The Information Gain of Features. ( $n$=2)

| Features | Benign Sample Set | | Malicious Sample Set | | Information Gain |
|---|---|---|---|---|---|
| | $y_i$=1 | $y_i$=0 | $y_i$=1 | $y_i$=0 | |
| FF84 | 98 | 11 | 92 | 0 | 0.000954615 |
| 4508 | 106 | 3 | 85 | 7 | 0.000387123 |
| FDFF | 109 | 0 | 89 | 3 | 0.001103624 |
| F33C | 101 | 8 | 76 | 16 | 0.002371356 |
| BF28 | 94 | 15 | 91 | 1 | 0.000767842 |
| … | … | … | … | … | … |

### 2.3 Using Rough Set Theory to Reduce Feature

Because the n-grams obtained from the method above is redundant, we should firstly reduce the feature dimension of our detection model as a preprocessing step. Here we use RST to reduce the feature dimension.

As we know that Rough Set Theory (RST) offers two fundamental concepts to deal with this particular problem: reduct and core. These concepts are generalized to families of equivalence relations. Some algorithms have been proposed to calculate reducts and core. Based on discernibility matrix and functions an algorithm to obtain core and reducts is defined [7]. However these routines are space and time consuming in practice for real world databases, thus, the need for an alternative implementation to improve and make more efficient large database processing. In this paper an efficient implementation to calculate relative core, based on positive region definition is presented. Our method of implementation takes into account the following considerations:

When checking if an equivalence relation $R \in P$ is $Q$-indispensable, where $P$ and $Q$ are families of equivalence relations $POS_{IND(P)}(IND(Q))$ is considered as a set of equivalence classes. In this case, indispensability is verified based on comparisons within the respective equivalence classes.

Verification is done in an incremental way, thus reducing calculation time when the equivalence relation is $Q$-indispensable. Comparisons between sets are time-consuming if we do it in a naive form -that is, each element at a time. When the positive region is considered as a set of equivalence classes, some properties are satisfied so that comparison within sets, then the calculation could be simplified based on its cardinalities. In our experiments an algorithm to calculate reducts based on the improved method is detailed in alg1.

### 2.4 Classification Method

Malicious code detection can be look as a binary classification problem. We use K Nearest Neighbor (KNN) and Support Vector Machine (SVM) classifiers to conduct the detection. The proposed detection procedure details as follow steps: 1) Dumping Hex-decimal byte sequence from malicious and benign PE format files. 2) Slicing each Hex sequence into gram by n –the size of sliding window. 3) Selecting feature based on

the Information Gain of each feature. 4) Reducing the feature dimension by using Rough Set Theory. 5) Training and checking the classifiers. And the detailed step is shown in figure 2.

```
Reducts( P,Q )
Input:  P, Q families of equivalence relations.
Output: R, R is a Q-reduct of P

1  Call  Core(P,Q)
2  R ← {R ∈ P | R is not marked}
3   if  R = ∅  then
4         return {P}
5   else
6         return  ∪ {Reducts(P − {R}), Q)}
                  R∈P
7   end

Procedure Core( P,Q )
input:   P, Q families of equivalence relations
Output:  R ∈ P , Q-indispensable are marked
8    indP ← U/IND(P)
9    indQ ← U/IND(Q)
10   for  ∀ W ∈ IND(Q)  do
11        m ← LowerCard(U / IND(P), W)
12        for   ∀ R ∈ P , not marked  do
13             n ← LowerCard(U / IND(P − {R}), W)
14              if  m ≠ n  then
15                 mark R
16              end
17        end
18   end

Procedure LowerCard( P,W )
Input: P partition ,  W ⊆ U .
Output: |IND(P)W|
19   cardinality ← 0
20   for  ∀S ∈ P  do
21         if  S ⊆ W  then
22              cardinality ← cardinality + |S|
23         end
24   end
25   return cardinality
```

**Alg 1.** Feature reduction Algorithm

In reality, it is likely to be impossible to collect all normal variations in behavior, so we must face the possibility that our normal database will provide incomplete coverage of normal behavior. If the normal were incomplete, false positives could be the result. Moreover, the inaccuracy of each classifier itself also needs to set some judge rules to improve the performance of the detecting systems. So our method is preliminary.

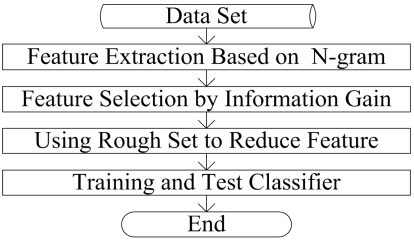


**Fig. 2.** Detection Procedure

## 3 Experiments

We collected 201 distinct Windows executable files (92 malicious code, and 109 benign codes) and labeled by a commercial virus scanner with the correct class label (malicious or benign) for our method. The total malicious code size is 3,614,242 byte, and benign code 5,765,650 bytes.

During the experiment, we use the software- Ngrams[8], LIBSVM[9] and RSES[10]. Ngrams tool is used in building byte n-gram profiles with parameters $8bit \le n \le 16bit$, and $20 \le L \le 4000$, where L is the number of features. RSES and LIBSVM tools are used to reduce feature and conduct classification. To evaluate our system we were interested in several quantities: False Negative (FN), the number of malicious executable examples classified as benign; False Positives (FP), the number of benign programs classified as malicious executables.

The experiment result shows in table 3. The results are very encouraging- for SVM classifier, it achieving accuracy of about 96% for several parameter configurations. The result shows a fact that the dataset can be represented as two 2-gram profiles, and be successfully used in the classification.

In another experiment [11], we had used a method based on Fuzzy Pattern Recognition algorithm (FPR) to classify virus. That algorithm had the lowest false positive

**Table 3.** Experiment Results

| Number of features | | KNN(%) | | SVM(%) | |
|---|---|---|---|---|---|
| Before Reduction | After Reduction | FN rate | FP rate | FN rate | FP rate |
| 20 | 17 | 18.48 | 13.04 | 13.76 | 10.87 |
| 50 | 39 | 19.57 | 15.22 | 13.76 | 11.96 |
| 100 | 76 | 11.96 | 10.87 | 7.34 | 7.61 |
| 200 | 153 | 9.78 | 9.78 | 5.50 | 6.52 |
| 500 | 385 | 8.70 | 7.61 | *4.59* | *3.26* |
| 1000 | 769 | 9.78 | 8.70 | 4.59 | 5.43 |
| 1500 | 1071 | 8.70 | 8.70 | 5.50 | 4.35 |
| 2000 | 1428 | 10.87 | 7.61 | 5.50 | 5.43 |
| 3000 | 2140 | 9.87 | 9.78 | 6.42 | 6.52 |
| 4000 | 2669 | 9.87 | 8.70 | 5.50 | 5.43 |

rate, 4.45%. The present method has the lowest false positive rate, 3.26%, which has better detection rates than the algorithm based on FPR. Notice that the detection rates of these two methods are nearly equal, but the FPR algorithm uses more training samples than the present method. This shows that our method is fit to detect malicious code when the gather of samples is difficult.

## 4 Conclusion

The present paper proposes the use of n-gram analysis in the area of anti-virus to make it more suitable as on-line detection system. In order to have a faster response from scanner, number of features should be minimized without affecting the classification power of the system. In the experiment we use rough set theory to reduce features that discard redundant attribute values. Experiment result shows that the present method could effectively use to discriminate benign and malicious PE format programs. The method achieves high rate of accuracy. Future work include testing this method over a larger set of malicious and benign executables for a fully evaluation of it. In addition with a larger data set, we plan to evaluate this method on different types of malicious codes such as Macros and Visual Basic scripts.

## Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grant No.60373023 and the Scientific Research Fund of Hunan Provincial Education Department of China under Grant No.05B072.

## References

1. Kephart, J., Arnold, W.: Automatic Extraction of Computer Virus Signatures. In: Proceedings.of the 4th Virus Bulletin International Conference, Abingdon, pp. 178–184 (1994)
2. Lo, R., Levitt, K., Olsson, R.: MCF: A Malicious Code Filter. Computers and Security 14, 541–566 (1995)
3. Tesauro, G., Kephart, J., Sorkin, G.: Neural networks for computer virus recognition. IEEE Expert. 8, 5–6 (1996)
4. Schultz, M., Eskin, E., Zadok, E., Stolfo, S.: Data mining methods for detection of new malicious executables. In: Proceedings of the, IEEE Symposium on Security and Privacy, Los Alamitos, pp. 38–49 (2001)
5. Kephart, J.: A Biologically Inspired Immune System for Computers, In: Proceedings of the Fourth International Workshop on Synthesis and Simulation of Living Systems, Massachusetts, pp. 130–139 (1994)
6. Damashek, M.: Gauging similarity with n-grams: language independent categorization of text. Science. 267, 843–848 (1995)
7. Skowron, A., Rauszer, C. (eds.): Intelligent decision support: Handbook of applications and advances of the Rough Set Theory. Kluwer Academic Publishers, Boston (1992)
8. Perl package Text::Ngrams: http://search. cpan. org /author /vlado /Text-Ngrams-0.03 N grams.pm

9. LIBSVM Tools Home Page: http://www.csie.ntu.edu.tw/ cjlin/
10. RSES Tools Home Page: http://logic.mimuw.edu.pl/ rses
11. Zhang, BY., Yin, JP., Hao, JB.: Using Fuzzy Pattern Recognition to Detect Unknown Malicious Executables Code. In: Wang, L., Jin, Y. (eds.) Fuzzy Systems and Knowledge Discovery. LNCS (LNAI), vol. 3613, pp. 629–634. Springer, Heidelberg (2005)