N-gram-based Detection of New Malicious Code

Tony Abou-Assaleh, Nick Cercone, Vlado Kešelj, Ray Sweidan
Privacy and Security Laboratory, Faculty of Computer Science, Dalhousie University
E-mail: {taa,nick,vlado,sweidan}@cs.dal.ca

Abstract

The current commercial anti-virus software detects a virus only after the virus has appeared and caused damage. Motivated by the standard signature-based technique for detecting viruses, and a recent successful text classification method, we explore the idea of automatically detecting new malicious code using the collected dataset of the benign and malicious code. We obtained accuracy of 100% in the training data, and 98% in 3-fold cross-validation.

1 Introduction

Since the appearance of the first computer virus in 1986, a significant number of new viruses has appeared every year.[1] 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 [3]. Even without this threat, the traditional approach consists of waiting for a number of computers to be infected, detecting the virus, designing a solution, and delivering and deploying the solution. A significant damage is done during this process. To address this problem, 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.[2]

The term *virus* is commonly used for malicious code, but for clarity reasons, we will use the term *malicious code* (MC) in further discussion, since it is relevant for all kinds of malicious code, such as viruses, worms, and Trojan horses. Since specific substrings of MC, or signatures, are typically used for MC detection, it is natural to use sets of such substrings as input to an automatic system. *N-grams*—all file substrings a fixed length n—are a good candidate for such a substring set. The idea of using n-grams for MC analysis is not new ([3] and [4] in 1994), but we did not find many reported results.

The *word* n-gram analysis was used successfully in language modeling and speech recognition [2], but character n-grams were only sparsely used. In 1994, character n-grams were used for text categorization in [1] with modest results. The Common N-Gram analysis (CNG) method [5] for text classification has been recently successfully used in automatic authorship attribution [5], text clustering, etc. Since n-grams overlap, they do not capture just statistics about substrings of length n, but they implicitly capture frequencies of longer substrings as well. This was noted in NLP, where tri-grams frequently perform very well even though they seem to be too short to capture any significant information. As a growing number of malicious code writers use tools to write and compile their code, n-grams could detect features of the code that are specific to certain tools, or families of tools, including code generators, compilers, and programming environments. In addition, n-grams could capture features that are specific for authors, coding styles, or even behavioural features. Since the captured features are implicit in the extracted n-grams, it would be difficult for virus writers to deliberately write viruses that fool n-gram analysis even when they have full access to the detection algorithm.

2 CNG Classification Method

The CNG method relies on profiles for class representation. After collecting n-grams from training date, the L most frequent n-grams with their normalized frequencies represent a class profile. The profile parameters are: the n-gram size n, and the profile length L. A new instance is classified by building its profile in the same way, and by using the kNN algorithm with $k = 1$. The following distance measure is used:

$$\sum_{s \in profiles} \left(\frac{f_1(s) - f_2(s)}{\frac{f_1(s)+f_2(s)}{2}} \right)^2 \quad (1)$$

[1] "Virus Writers: The End of The Innocence?" by Sarah Gordon, IBM Thomas J. Watson Research Center, http://www.research.ibm.com/antivirus/SciPapers/VB2000SG.htm The WildList — http://www.wildlist.org/

[2] A criterion for detecting viruses may be adapted to specific users. For some users any executable attachment in an e-mail message can be immediately classified as malicious code, while other users do exchange executable code by e-mail.

Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04)
0730-3157/04 $20.00 © 2004 **IEEE**

L	n									
	1	2	3	4	5	6	7	8	9	10
20	0.71	0.68	0.62	0.82	0.80	0.89	0.91	0.85	0.92	0.89
50	0.89	0.77	0.65	0.95	0.88	0.88	0.85	0.88	0.92	0.94
100	0.92	0.95	0.80	0.97	0.92	0.92	0.92	0.94	0.88	0.94
200	0.94	0.97	0.94	0.89	0.95	0.95	0.97	0.97	0.97	0.95
500	0.94	0.95	0.94	0.85	0.95	0.97	0.92	0.88	0.91	0.92
1000	0.94	0.97	0.98	0.97	0.98	0.98	0.98	0.97	0.97	0.97
1500	0.94	0.95	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
2000	0.94	0.95	0.98	0.98	0.98	0.98	0.98	1.00	0.98	1.00
3000	0.94	0.95	0.98	0.98	0.98	0.98	1.00	1.00	1.00	1.00
4000	0.94	0.95	0.98	0.98	0.98	1.00	1.00	1.00	0.98	0.98
5000	0.94	0.97	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98

Table 1. Training accuracy

L	n									
	1	2	3	4	5	6	7	8	9	10
20	0.71	0.77	0.70	0.79	0.81	0.80	0.77	0.82	0.85	0.85
50	0.88	0.76	0.73	0.86	0.88	0.74	0.83	0.82	0.85	0.83
100	0.91	0.88	0.74	0.77	0.88	0.83	0.89	0.88	0.83	0.89
200	0.91	0.94	0.76	0.76	0.91	0.91	0.91	0.88	0.88	0.86
500	0.79	0.97	0.89	0.76	0.92	0.95	0.89	0.86	0.89	0.88
1000	0.79	0.97	0.97	0.94	0.97	0.95	0.92	0.88	0.92	0.94
1500	0.79	0.95	0.98	0.97	0.98	0.97	0.94	0.91	0.91	0.94
2000	0.79	0.92	0.98	0.97	0.98	0.98	0.94	0.94	0.95	0.95
3000	0.79	0.94	0.98	0.97	0.97	0.97	0.97	0.95	0.97	0.98
4000	0.79	0.92	0.97	0.97	0.97	0.97	0.98	0.97	0.97	0.97
5000	0.79	0.92	0.97	0.95	0.94	0.95	0.98	0.98	0.97	0.97

Table 2. 3-fold cross-validation accuracy

where s is any n-gram from one of the two profiles, and $f_i(s)$ are n-gram frequencies in two profiles. The frequency difference is divided by $(f_1(s) + f_2(s))/2$ to balance the weight between low- and high-frequent n-grams [5].

3 Experimental Results

We collected 65 distinct Windows executable files (25 MC, and 40 benign code (BC)), extracted from e-mail messages, of size from 12.5 to 420 KB. The total MC size is 831KB, and BC 5.5MB. Ngrams [6] tool is used in building byte n-gram profiles with parameters $1 \leq n \leq 10$, and $20 \leq L \leq 5000$.

Experiment 1: Training error. The MC and BC profiles are built using all available code. Each file is then classified as MC or BC using the CNG method, and classification accuracy is measured for different combinations of parameter values. The results (Table 1) are very encouraging, achieving accuracy of 100% for several parameter configurations. A high accuracy of 98% and more is achieved for $n \geq 3$ and $L \geq 1500$. This is a biased evaluation experiment, since the training data is used in testing, however the result shows an important fact that the 5MB of BC and 0.8MB of MC can be represented as two 1500-length tri-gram profiles (of size 10.5KB) with a very simple algorithm, and be successfully used in the classification.

Experiment 2: 3-fold cross-validation. To obtain unbiased evaluation results, we performed 3-fold cross-validation [7]. The data is randomly partitioned into 3 disjoint datasets or folds. Two of these datasets are used for training and the remaining dataset is used for testing. The process is repeated 3 times, each time using a different testing dataset. The number of files is not large, so we choose 3-fold cross-validation, instead of a larger number of folds. The folds are created in random, balanced way, i.e., approximately 1/3 of worm files and 1/3 of benign files are selected in each fold. The resulting average accuracy (Table 2) provides more positive evidence for the use of the CNG method in the MC detection. The average accuracy is high, achieving 98% for several parameter configurations.

4 Conclusions and Future Work

We have demonstrated encouraging preliminary results in applying the CNG method based on byte n-gram analysis in the detection of malicious code. The method achieves 100% accuracy on training data, and 98% accuracy in 3-fold cross-validation. The future work includes experiments on larger data collection, mining the extracted n-grams to refine the method for extraction of the MC signatures, and experiments with reverse-engineered MC source code.

References

[1] W. Cavnar and J. Trenkle. 1994. "N-gram-based text categorization." In *Proceedings SDAIR-94*.

[2] D. Jurafsky and H. M. James. 2000. *Speech and Language Processing*. Prentice-Hall, Inc.

[3] J.O. Kephart and W.C. Arnold. 1994. "Automatic Extraction of Computer Virus Signatures." In *Proc.of the 4th Virus Bulletin Int'l Conf.*. Virus Bulletin Ltd., Abingdon, pp. 178-184.

[4] J.O. Kephart. 1994. "A Biologically Inspired Immune System for Computers." In *Artificial Life IV, Proc.of the 4th In'l Wsh. on Synthesis and Simulation of Living Systems*. MIT Press, pp. 130-139.

[5] V. Kešelj, F. Peng, N. Cercone, and C. Thomas. 2003. "N-gram-based Author Profiles for Authorship Attribution." In *Proc.of PACLING'03*, Halifax.

[6] V. Kešelj. 2003–04. Perl package Text::Ngrams. WWW: http://search. cpan.org/author/VLADO/Text-Ngrams-0.03/Ngrams.pm.

[7] C. Manning and H. Schuetze. 1999. Foundations of Statistical Natural Language Processing. The MIT Press.