Detection of New Malicious Code Using N-grams Signatures

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

***Abstract*— Signature-based malicious code detection is the standard technique in all commercial anti-virus software. This method can detect a virus only after the virus has appeared and caused damage. Signature-based detection performs poorly when attempting to identify new viruses. Motivated by the standard signature-based technique for detecting viruses, and a recent successful text classification method, n-grams analysis, we explore the idea of automatically detecting new malicious code. We employ n-grams analysis to automatically generate signatures from malicious and benign software collections. The n-grams-based signatures are capable of classifying unseen benign and malicious code. The datasets used are large compared to earlier applications of n-grams analysis.**

I. 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 [5]. 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 in further discussion, since it is relevant for all kinds of malicious code, such as viruses, worms, and Trojan horses.

Since the specific substrings of malicious code (MC), or signatures, are typically used to detect certain type of MC, it is natural to use sets of such substrings as indicators for MC detection. *N-grams*—all substrings of a file of a fixed length n—are a candidate for such a set that can be efficiently collected. The idea of using n-grams for MC analysis is not new. In 1994, a byte n-gram-based method for automatic extraction of virus signatures was described in [5]. Similarly, an n-gram-based method is used in a proposal for a "computer immune system" in [6]. However, there is little literature on this approach after 1994.

The *word* n-gram analysis, which uses a window of n consecutive words, has been used for a while in natural language processing (NLP). For example, it is successfully used in language modelling and speech recognition [4]. On the other hand, the *character* n-gram analysis was only sparsely used. In 1994, character n-grams were used for text categorization in [3] with modest results. The Common N-Gram analysis (CNG) method [7] for text classification has been recently successfully used in automatic authorship attribution [7], detection of dementia of Alzheimer's type [8], and text clustering. The CNG method was motivated by an approach introduced by W. R. Bennett in 1976 [2], in which the letter bi-gram frequencies were used in authorship attribution.

Byte n-grams of a file are overlapping substrings, collected in a sliding-window fashion where the windows of size n slides one byte at a time. Concordantly, they do not capture just statistics about substrings of length n, but they implicitly capture frequencies of longer substrings as well. This phenomenon was noted in NLP, where tri-grams frequently perform very well even though they seem to be too short to capture any significant information. N-grams have the ability to capture implicit features of the input that are difficult to detect explicitly. As a growing number of MC writers use tools to write and compile their code, n-grams could detect features of the code that are specific for certain tools, or families of tools, including code generators, compilers, and programming environment. In addition, n-grams could capture features that are specific for authors, coding styles, or even behavioural features.

Given a database of MC and benign code (BC), n-gram analysis can be used to extract the most frequent n-grams, which act as signatures. When a new code is analyzed, it can be classified as benign or malicious based on the category that it matches the most. Thus, n-grams could predict the maliciousness of unseen code and capture new viruses that share features with previously learned viruses. Since the cap-

[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 adopted to a specific user. For some users any executable attachment in an e-mail message can be immediately classifies as malicious code, while other users do exchange executable code by e-mail.

TABLE I

WIN32 COLLECTION: ACCURACY WITH A LIMIT OF $100,000$

	n														
L	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
20	.45	.58	.52	.63	.69	.57	.55	.51	.50	.49	.44	.42	.38	.36	.37
50	.60	.63	.88	.89	.87	.84	.73	.68	.82	.63	.79	.79	.82	.85	.82
100	.77	.75	.90	.87	.87	.89	.87	.84	.84	.86	.86	.87	.87	.86	.85
200	.85	.76	.87	.89	.90	.90	.92	.89	.90	.90	.90	.89	.89	.89	.90
500	.85	.89	.89	.91	.90	.90	.91	.90	.88	.88	.87	.87	.84	.85	.85
1K	.85	.92	.93	.93	.91	.90	.89	.87	.85	.84	.84	.83	.85	.80	.80
2K	.85	.87	.93	.92	.90	.87	.86	.84	.83	.81	.83	.83	.86	.79	.75
3K	.85	.83	.92	.91	.90	.87	.86	.83	.82	.81	.81	.82	.84	.93	.93
4K	.85	.78	.91	.91	.89	.87	.85	.82	.81	.85	.92	.92	.92	.92	.92
5K	.85	.73	.91	.89	.89	.87	.84	.81	.92	.93	.93	.92	.92	.92	.92
6K	.85	.69	.91	.88	.88	.86	.84	.92	.93	.93	.93	.92	.92	.92	.92
7K	.85	.65	.91	.87	.87	.84	.92	.92	.93	.93	.93	.92	.92	.92	.92
8K	.85	.63	.91	.87	.87	.84	.92	.92	.93	.93	.93	.92	.92	.92	.92
9K	.85	.61	.90	.86	.87	.92	.92	.92	.93	.93	.93	.92	.92	.92	.92
10K	.85	.59	.89	.86	.86	.92	.92	.92	.93	.93	.93	.92	.92	.92	.92

TABLE II

WIN32 COLLECTION: ACCURACY WITH A LIMIT OF $200,000$

	n														
L	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
20	.45	.59	.50	.62	.70	.60	.55	.53	.51	.50	.46	.44	.41	.40	.42
50	.60	.64	.88	.89	.87	.84	.73	.69	.81	.63	.78	.79	.83	.85	.82
100	.77	.75	.90	.88	.88	.91	.87	.85	.84	.85	.86	.87	.87	.87	.85
200	.85	.76	.87	.89	.91	.91	.93	.90	.89	.90	.90	.90	.89	.89	.90
500	.85	.90	.88	.91	.91	.90	.91	.91	.89	.89	.88	.87	.84	.84	.85
1K	.85	.92	.93	.93	.92	.90	.90	.88	.88	.87	.87	.84	.84	.84	.85
2K	.85	.90	.94	.92	.91	.89	.89	.86	.86	.84	.83	.82	.83	.82	.81
3K	.85	.83	.93	.91	.90	.87	.85	.85	.84	.83	.82	.82	.82	.78	.77
4K	.85	.82	.92	.91	.90	.86	.85	.85	.83	.82	.81	.81	.81	.76	.75
5K	.85	.78	.93	.91	.89	.86	.85	.84	.83	.80	.78	.78	.78	.74	.73
6K	.85	.75	.93	.88	.88	.86	.85	.82	.80	.79	.76	.75	.76	.73	.72
7K	.85	.72	.92	.87	.87	.86	.84	.81	.79	.78	.76	.74	.74	.72	.72
8K	.85	.70	.90	.86	.87	.86	.84	.81	.79	.77	.75	.74	.74	.71	.92
9K	.85	.69	.90	.86	.86	.85	.83	.81	.79	.93	.75	.93	.92	.92	.92
10K	.85	.66	.90	.86	.85	.85	.82	.80	.94	.93	.92	.93	.92	.92	.92

tured 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.

The datasets used in our experiments are in the order of tens of megabytes in size, which is small compared to the terabyte virus repositories, but is considerably larger than the datasets traditionally used in n-grams analysis, which usually does not exceed a few megabytes. In addition to using n-grams analysis for malicious code detection, this work presents

II. CNG METHOD

The CNG classification method relies on profiles for class representation. During training, the data for each class is collected and n-grams with their normalized frequencies are counted. The L most frequent n-grams with their normalized frequencies represent a class profile. There are two parameters in building a profile: n — the n-gram size, and L — the profile length. When a new instance needs to be classified, the instance profile is built in the same way. The instance is classified using k-nearest neighbour algorithm with $k = 1$; i.e., the similarity distance is measured between the instance profile and class profiles, and the class with the closest class profile is chosen. The following distance measure is used:

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

where s is any n-gram from one of the two profiles, $f_1(s)$ is frequency of the n-gram in one profile, or 0 if the n-gram does not exist in the profile, and $f_2(s)$ is the frequency of the n-gram in another profile. The difference between frequencies is divided by $(f_1(s)+f_2(s))/2$ in order to make the difference relative instead of absolute, so that the same weight is given to the difference between low-frequent n-grams as for the high-frequent n-grams.

III. EXPERIMENTAL RESULTS

In our earlier experiments [1], we used a small collection of 25 worms (831KB) that we extracted from infected email messages and 40 *healthy* Windows executable files (5.5MB). We achieved a training accuracy of 100% for several parameter configurations and a 3-fold cross-validation average accuracy of 98%.

Following the encouraging results of using the CNG method with the small worm collection, we conducted series of experiments with two larger collections of MC: the *I-Worm* collection and the *Win32* collection. These collections are available from [11]. The I-Worm collection consists of 292 Internet worms that are Windows binary executable files. The total size of the I-Worm collection is 15.7MB. The Win32 collection contains 493 Windows binary executable viruses whose names begin with "Win32"; their total size is 21.2MB.

There are computational issues that must be considered when dealing with larger code collections. In our initial experiments, we computed n-grams of sizes up to 10 bytes. The number of potential n-grams of size 10 is $2^{10*8} \approx 10^{24}$. The small code size in our initial experiments resulted in a computationally-tractable upper bound on the number of possible n-grams. With the larger code collections, computing the frequencies of all seen n-grams is impractical. The Perl software tool Ngrams [9], which we use in our experiments, supports the *limit* parameter; if the number of collected n-grams exceeds twice the limit, the list of n-grams is reduced to the limit by removing the least frequent n-grams from the list.

A. Parameter selection

There are three primary parameters in our experiments: the limit, the maximum length of the n-grams, and the maximum profile size. The limit ensures that the computation does not overflow the physical memory and is a single-valued parameter. The length of the n-grams, n, takes values from 1 to the specified maximum. As evident in table I and table II,

TABLE III

I-WORM COLLECTION: TRAINING ACCURACY FOR DIFFERENT VALUES OF N-GRAM SIZE (n) AND PROFILE SIZE (L)

	n									
L	1	2	3	4	5	6	7	8	9	10
20	0.54	0.50	0.65	0.74	0.68	0.64	0.52	0.50	0.52	0.43
50	0.62	0.62	0.83	0.80	0.85	0.83	0.72	0.65	0.60	0.57
100	0.80	0.65	0.76	0.68	0.84	0.86	0.85	0.83	0.83	0.85
200	0.75	0.69	0.63	0.62	0.79	0.86	0.89	0.87	0.89	0.88
500	0.57	0.87	0.88	0.70	0.83	0.89	0.88	0.87	0.88	0.89
1000	0.57	0.85	0.89	0.90	0.90	0.89	0.88	0.88	0.89	0.87
1500	0.57	0.86	0.89	0.91	0.88	0.90	0.86	0.85	0.83	0.84
2000	0.57	0.83	0.90	0.90	0.88	0.87	0.84	0.79	0.73	0.74
3000	0.57	0.81	0.88	0.89	0.86	0.83	0.71	0.71	0.64	0.65
4000	0.57	0.78	0.88	0.87	0.84	0.82	0.68	0.64	0.61	0.62
5000	0.57	0.76	0.88	0.85	0.80	0.80	0.64	0.62	0.58	0.61

TABLE IV

WIN32 COLLECTION: TRAINING ACCURACY FOR DIFFERENT VALUES OF N-GRAM SIZE (n) AND PROFILE SIZE (L)

	n									
L	1	2	3	4	5	6	7	8	9	10
20	0.45	0.59	0.51	0.63	0.67	0.59	0.54	0.52	0.51	0.47
50	0.60	0.63	0.88	0.88	0.87	0.85	0.74	0.68	0.81	0.64
100	0.76	0.73	0.90	0.88	0.87	0.90	0.87	0.85	0.84	0.85
200	0.85	0.74	0.87	0.89	0.92	0.90	0.93	0.89	0.89	0.90
500	0.85	0.87	0.89	0.91	0.90	0.90	0.91	0.91	0.90	0.89
1000	0.85	0.90	0.93	0.93	0.91	0.90	0.89	0.88	0.87	0.87
1500	0.85	0.89	0.94	0.94	0.91	0.89	0.88	0.87	0.87	0.86
2000	0.85	0.87	0.94	0.92	0.91	0.89	0.87	0.86	0.85	0.82
3000	0.85	0.84	0.93	0.91	0.90	0.86	0.83	0.81	0.80	0.80
4000	0.85	0.79	0.93	0.92	0.87	0.86	0.81	0.80	0.80	0.79
5000	0.85	0.75	0.93	0.91	0.87	0.86	0.81	0.80	0.78	0.78

larger values of n do not always result in a better performance. The maximum value of n should be chosen large enough as to demonstrate that the optimal value of n is withing the tested range. The profile size, L, determines the number of the most frequent n-grams of a file that are used as a signature for that file. Similar to n, larger values of L do not always result in a better performance and a maximum value should be chosen as to encompass the optimal value of L.

In our experience with n-grams, we found that a limit of $100,000$, n-grams of lengths 1 to 10, and profile sizes ranging from 20 to $5,000$ most frequent n-grams provided good results. We conducted several runs to verify whether these choices of values are suitable for the I-Worm collection and Win32 collection. The results of two runs using the Win32 collection are shown in table I and table II. In these two runs, we increased the maximum n-grams length to 15 and the maximum profile size to $10,000$. Table I shows the results with a limit of $100,000$ n-grams and table II shows the results with a limit of $200,000$ n-grams.

We did not find a significant improvement in training accuracy when increasing the limit to $200,000$, the maximum n-grams length to 15, or the maximum profile size to $10,000$. Therefore, all the experiments described below are configured with a limit of $100,000$, a maximum n-grams length of 10, and a maximum profile size of $5,000$.

B. Training accuracy

In the training accuracy experiments, we build a *malicious profile* using all available MC, and, in a similar fashion, a *benign profile* is built from the BC. Each file is then classified as a malicious or a benign program using the CNG method, and we measured the accuracy for different combinations of parameters n (n-gram size) and L (profile size). The results of the I-Worm collection and the Win32 collection are shown in table III and table IV, respectively.

The results are very encouraging, achieving accuracy of over 90% for several parameter configurations. An accuracy of and 91% for $n = 4$ and $L = 1500$ is achieved for the I-Worm collection, and 94% for the Win32 collection using the same parameters, as well as others. This is a biased evaluation experiment, since the same training data is used in testing. Even though this is a biased experiment, the result is significant since it shows that the 5MB corpus of BC, 15.7MB of I-Worm MC, and 21.2 of Win32 MC can be represented as 1500-length quad-gram profiles with a very simple algorithm, and be successfully used in the classification.

TABLE V

I-WORM COLLECTION: AVERAGE ACCURACY IN 5-FOLD CROSS-VALIDATION FOR DIFFERENT VALUES OF N-GRAM SIZE (n) AND PROFILE SIZE (L)

	n									
L	1	2	3	4	5	6	7	8	9	10
20	0.59	0.49	0.61	0.64	0.72	0.64	0.57	0.49	0.50	0.45
50	0.67	0.55	0.80	0.76	0.83	0.83	0.63	0.58	0.60	0.55
100	0.81	0.73	0.72	0.73	0.70	0.84	0.81	0.79	0.81	0.82
200	0.77	0.69	0.69	0.66	0.79	0.85	0.86	0.85	0.87	0.86
500	0.56	0.85	0.85	0.78	0.82	0.86	0.87	0.85	0.86	0.86
1000	0.56	0.84	0.89	0.89	0.90	0.88	0.85	0.87	0.88	0.87
1500	0.56	0.82	0.89	0.91	0.88	0.89	0.87	0.85	0.84	0.85
2000	0.56	0.83	0.89	0.89	0.87	0.87	0.85	0.83	0.82	0.84
3000	0.56	0.82	0.88	0.88	0.87	0.84	0.80	0.82	0.80	0.81
4000	0.56	0.80	0.87	0.86	0.83	0.81	0.79	0.81	0.78	0.80
5000	0.56	0.79	0.86	0.83	0.81	0.81	0.79	0.79	0.77	0.78

C. 5-fold cross-validation

To obtain unbiased evaluation results, we performed a 5-fold cross-validation. In the cross-validation method [10], the data is randomly partitioned into n disjoint datasets or folds. $n-1$ of these datasets are used for training and the remaining dataset is used for testing. The process is repeated n times, each time using a different testing dataset. The results in these n evaluations are averaged to obtain the final result.

The folds are created in a random, balanced way, i.e., approximately 1/5 of malicious files and 1/5 of benign files are selected in each fold. The average accuracy using the I-Worm collection and the Win32 collection are shown in table V and table VI, respectively.

The result provides more positive evidence for the use of the CNG method in the MC detection. The average accuracy is high, achieving 91% for both the I-Worm collection and the

TABLE VI

WIN32 COLLECTION: AVERAGE ACCURACY IN 5-FOLD CROSS-VALIDATION FOR DIFFERENT VALUES OF N-GRAM SIZE (n) AND PROFILE SIZE (L)

	n									
L	1	2	3	4	5	6	7	8	9	10
20	0.64	0.63	0.63	0.61	0.58	0.58	0.55	0.52	0.50	0.47
50	0.58	0.70	0.81	0.87	0.85	0.86	0.80	0.63	0.68	0.64
100	0.75	0.74	0.90	0.87	0.87	0.89	0.88	0.85	0.86	0.85
200	0.85	0.70	0.87	0.88	0.90	0.90	0.91	0.88	0.87	0.89
500	0.85	0.81	0.88	0.91	0.90	0.90	0.90	0.89	0.89	0.88
1000	0.85	0.88	0.90	0.91	0.89	0.89	0.86	0.86	0.87	0.86
1500	0.85	0.86	0.91	0.91	0.90	0.88	0.87	0.87	0.87	0.85
2000	0.85	0.86	0.91	0.91	0.89	0.88	0.87	0.85	0.84	0.84
3000	0.85	0.84	0.91	0.90	0.88	0.87	0.85	0.84	0.83	0.83
4000	0.85	0.84	0.91	0.91	0.89	0.86	0.86	0.84	0.82	0.82
5000	0.85	0.79	0.91	0.90	0.88	0.86	0.86	0.83	0.81	0.87

Win32 collection.

IV. CONCLUSIONS AND FUTURE WORK

We have demonstrated encouraging initial results in applying the CNG method based on byte n-gram analysis in the detection of MC. We used two datasets of MC, I-Worm collection and Win32 collection, and a collection of BC. The size of the MC code of each of the collections is considerably larger than what is traditionally used in n-grams analysis, and still satisfactory results are obtained using the same parameter values for n and L as in other works. The method achieves over 90% accuracy on training data using each of the two datasets, and 91% accuracy in 5-fold cross-validation. The future work includes experiments on larger data collection with sizes of the order of hundreds of megabytes. Currently, we use the CNG method as black box for detecting viruses and worms. Mining the extracted n-grams may help refine the extraction of the MC signatures. Experimenting with reverse-engineered MC source code is another direction that we plan to pursue.

ACKNOWLEDGMENT

This work is supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).

REFERENCES

[1] T. Abou-Assaleh, N. Cercone, V. Kešelj, and R. Sweidan. 2004. "N-gram-based Detection of New Malicious Code." In *Proceedings of the 24th Annual International Computer Software and Applications Conference (COMPSAC 2004)*, Hong Kong.

[2] W.R. Bennett. 1976. *Scientific and engineering problem-solving with the computer*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

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

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

[5] J.O. Kephart and W.C. Arnold. 1994. "Automatic Extraction of Computer Virus Signatures." In *Proceedings of the 4th Virus Bulletin International Conference*. R. Ford, ed., Virus Bulletin Ltd., Abingdon, England, pp. 178-184.

[6] J.O. Kephart. 1994. "A Biologically Inspired Immune System for Computers." In *Artificial Life IV, Proc.of the Fourth Intern.Workshop on Synthesis and Simulation of Living Systems*. Rodney A. Brooks and Pattie Maes, eds., MIT Press, Cambridge, Massachusetts, pp. 130-139.

[7] V. Kešelj, F. Peng, N. Cercone, and C. Thomas. 2003. "N-gram-based Author Profiles for Authorship Attribution." In *Proceedings of the Conference Pacific Association for Computational Linguistics, PACLING'03*, Dalhousie University, Halifax, Nova Scotia, Canada.

[8] V. Kešelj, E. Asp, K. Rockwood, and N. Cercone. 2003. "Computational analysis of language used by Alzheimer patients in interviews." In *Proceedings of the 6th Annual Symposium on the Treatment of Alzheimer's Disease*, Halifax.

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

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

[11] VX Heavens. 2004-06. WWW: http://vx.netlux.org/.