N-GRAMS-BASED FILE SIGNATURES FOR MALWARE DETECTION

Igor Santos, Yoseba K. Penya, Jaime Devesa and Pablo G. Bringas
S^3Lab, Deusto Technological Foundation, Bilbao, Basque Country
{isantos, ypenya, jdevesa, pgb}@tecnologico.deusto.es

Keywords: Security, computer viruses, data-mining, malware detection, machine learning.

Abstract: Malware is any malicious code that has the potential to harm any computer or network. The amount of malware is increasing faster every year and poses a serious security threat. Thus, malware detection is a critical topic in computer security. Currently, signature-based detection is the most extended method for detecting malware. Although this method is still used on most popular commercial computer antivirus software, it can only achieve detection once the virus has already caused damage and it is registered. Therefore, it fails to detect new malware. Applying a methodology proven successful in similar problem-domains, we propose the use of n-grams (every substring of a larger string, of a fixed lenght *n*) as file signatures in order to detect unknown malware whilst keeping low false positive ratio. We show that n-grams signatures provide an effective way to detect unknown malware.

1 INTRODUCTION

The term malware was coined to name any computer program with malicious intentions, such as viruses, worms, or Trojan horses. As one may think, parallel to the grow of the Internet, the amount, power, and variety of malware increases every year (Kaspersky, 2008), as well as its ability to avoid all kind of security barriers.

The classic method to detect these threats consists on waiting for a certain number of computers to be infected, determining then a file signature for the virus and finally finding a specific solution for it. In this way, based on the list of signatures (also known as signature database (Morley, 2001)), the malware detection software can provide protection against known viruses (ie. those on the list). This approach has proved to be effective when the threats are known in beforehand, and is the most extended solution within antivirus software. Still, as already mentioned, it fails when facing new ones.

Moreover, upon new virus apparition and until the corresponding file signature is obtained, mutations of the original virus released in the meanwhile may escape to detection based on that signature.

These facts have led to a situation in which malware writers develop new viruses and different ways for hiding their code, while researchers design new tools and strategies to detect them (Nachenberg, 1997). Such evolution makes very difficult to develop an universal malware detector. Thus, the task of the researcher is to achieve very good results against known malware and to turn the attempt of writing new undetectable virus more difficult.

Generally, there are several indicators to evaluate the effectiveness of a new malware detection system. First, we have to look at its malware detection ratio (i.e. the amount of viruses of a sample that the software detects). Second, we have to look at the false positive ratio (i.e. the amount of non malicious programs that are erroneously classified as malware), since it determines how practical the method is to be commercialised: a new system able to detect a lot of malware but at the cost of a high rate of false positives is not practical in the real world, where the system must deal with a lot of benign software that should not be classified as malware.

Therefore, antivirus companies usually prefer a modest detection ratio with low (or ideally zero) false positive ratio rather than a notable detection one with high also a false positive ratio.

Language recognition is a research area that has tackled a similar problem, since they also have to deal with the retrieval of information that is hard to see at the first glance. The most extended technique against this problem has been the use of the so-called n-grams. For instance, in Chinese document classification (Zhou and Guan, 2002) or text classification (Jacob and Gokhale, 2007).

N-grams are all substrings of a larger string with a length *n*. A string is simply split into substrings of fixed length *n*. For example, the string *"MALWARE"*, can be segmented into several 4-grams: *"MALW"*, *"ALWA"*, *"LWAR"*, *"WARE"* and so on.

Against this background, this paper advances the state of the art in two main ways. First, we address here a new methodology for malware detection based on the use of n-grams for file signatures creation. Second, we tackle the issue of dealing with false positives using a parameter named *d* to control how strict the method behaves to classify the instance as malware or benign software in order to avoid false positives.

The remainder of this paper is structured as follows. Section 2 discusses related work. Section 3 introduces the method proposed in this paper. Section 4 describes the experiments performed and discusses the obtained results. Finally, Section 5 concludes and outlines the avenues of future work.

2 RELATED WORK

Lately, the problem of detecting unknown malicious code has been addressed by using machine learning and data mining (Schultz et al., 2001). In their research they propose to use different data mining methods for detection of new malware, however, none of the techniques proposed had good balance between false positive ratio and malware detection ratio in their experimental results.

In a verge closer to our view, N-grams were used first for malware analysis by an IBM research group (Kephart, 1994). They proposed a method to automatically extract signatures for the malware. Still, there was no experimental results in their research.

Furthermore, Assaleh *et al* in (Abou-Assaleh et al., 2004) addressed an n-grams-based signature method to detect computer viruses. In that approach, given a set of non malicious programs and computer viruses code, n-grams profiles for each class of software (malicious or benign software) were generated, in order to later classify any unknown instance into benign or malicious code using a *k-nn algorithm* with $k = 1$. That experiment had very good results in terms of malware detection ratio; still, no false positive ratio was given in the experiment results. This lack of false positives in the experimental results and the absence of any technique to control the appearance of them renders this method to be unpractical in a commercial way.

3 METHOD DESCRIPTION

Our technique relies on a large set of training values in order to build representation for each file in that set. This set is composed of a collection of malware software and benign programs. Specifically, the malware is made up of different kind of malicious software (i.e. computer viruses, Trojan horses, spyware, etc).

Once the set is chosen, we extract n-grams for every file in that set that will act as the file signature. Hereafter, the system can classify any unknown instance as malware or benign software. To this extent, we classify the unknown instance using *k-nearest neighbour algorithm* (Fix and Hodges, 1952), one of the simplest machine learning algorithms that can be used in classifying issues. This algorithm relies on identifying the *k* most nearest (say most similar) instances, to later classify the unknown instance based on which class (malware or benign) are the k-nearest instances.

The following measure function is used in order to detect how much the unknown instance looks like the known ones:

$$\sum_{x \epsilon X} \frac{f(x)}{card(X) + card(Y)} \tag{1}$$

where X is the set of the n-grams of the unknown instance, x is any n-gram in the set X, Y is the set of n-grams of the instance its class is known, and $f(x)$ is the number of coincidences of the n-gram x in the set Y.

When every instance of the known set is measured comparing it with the unknown instance, we select the *k* highest values of the measured instance, and we consider the unknown instance as malware only if on the *k* most alike files the amount of malware instances minus the amount of benign instances is greater or equal than a parameter *d*, as shown in the following formula:

$$MW(K) - GW(K) >= d \tag{2}$$

where $MW(K)$ is the amount of malware instances in the *k* nearest neighbours, $GW(K)$ is the amount of benign software in the *k* nearest neighbours and *d* is the parameter d.

This parameter *d* controls how strict the system is going to be in order to classify the unknown instance as malware or benign software. Moreover, this parameter is what we are going to use to keep low the false positive ratio; with a high value of d (that has to be always lesser or equal than k), we predict that the false positive ratio is going to keep low.

4 EXPERIMENTAL RESULTS

In order to perform our experiments, a computer security software company provided us with a slice of a larger malware database. More accurately, it consisted of 149882 malware files and texts associated to them, and 4934 benign software with their texts. These texts have been extracted by several tools that provide information of the execution of the malware instances in a safe environment. In this way, these associated texts gave us a very valuable information of the behaviour of malware instances.

The huge size of the database (100 GB) makes the dataset to be nearly unpractical for research purposes; therefore, we decided to use a small portion of it for our experiments. In this way, we choose randomly 1000 malware files and 1000 benign software files to accomplish our experiments, having an equal quantity of malware and benign instances. This sample represents only a 1% of the original database, anyway the amount of the sample is large enough to show us if n-grams file signatures are able to achieve detection of unknown malware.

Further, we divided this dataset into a set of training (the known files) and a set for classifying (the unknown ones). This kind of division is usually performed in machine learning experiments. In this way, we train the model with enough information to build representation for capturing features of the behaviours of the malware. To this extent, we divide the sample into a set of training, that is going to be compound by the 66% of the total sample and a 34% of benign software.

With the training set we train the model for later test the model with the remaining 34%. The 66% to act as the training instances can train the model in a efficient way, as the 34% of the instances for testing are able to show the capability of the model for detecting new and unknown malware.

4.1 Parameters

There are three main parameters in our experiments: the size of the n-grams, the number of nearest neighbours for the knn algorithm and the limit that marks the nature of the unknown instance as malware (this value is over the half).

First, the size of the n-grams, or n, allows us to decide how long in bytes the n-gram will be. In the experiments presented here, we run tests with $n = 2$, $n = 4$, $n = 6$ and $n = 8$.

Second, k; the number of nearest neighbours for the knn algorithm. It is hard to establish an optimal value for this parameter, and it usually depends on the system's behaviour.

Finally, the parameter d, as aforementioned, establishes how strict is going to be the system for classifying a unknown instance as malware. One of our primary goals is to keep as low as possible the amount of false positives, therefore this parameter is going to take the same value as k in at least half of the tests run.

4.2 Results

In the experiments, we have built file signatures that are made up of the set of n-grams for every file in the sample (2000 files), for *n=2*, *n=4*, *n=6* and *n=8*. These files content n-grams of the texts retrieved from the executables. We split these files containing the file signatures into two separate files for every value of n. One of the two files contents the n-grams of the files that are going to be used as the training set, and the other one contents the n-grams of the files that are going to be used as the test set.

Figure 1 shows the obtained false positive and detection ratio over the parameters n, k and d. More accurately, for $n = 2$, the detection ratio is quite low, achieving a maximum detection ratio of 69.66%, thus 2-grams do not seem to be appropriate for file signatures. In this experiment we achieved best results for detection ratio for $n = 4$, getting a maximum detection ratio of 91.25 % for $k = 7$ and $d = 2$. For the following n values, detection ratio is lower than for $n = 4$, however, the second best results are achieved for a value for n of 8.

Besides, the false positive ratio and detection ratio themselves do not show the potential of the system. If we focus on the best results when false positive ratio is 0%, we achieve the best result for $n = 4$, $k = 17$ and $d = 17$, keeping a 74.37% detection ratio.

5 CONCLUSION AND FUTURE WORK

As the amount of malware and its variety is growing every year, malware detection rises to become a topic of research and concern. Further, classic signature methods provide detection when the threat is known

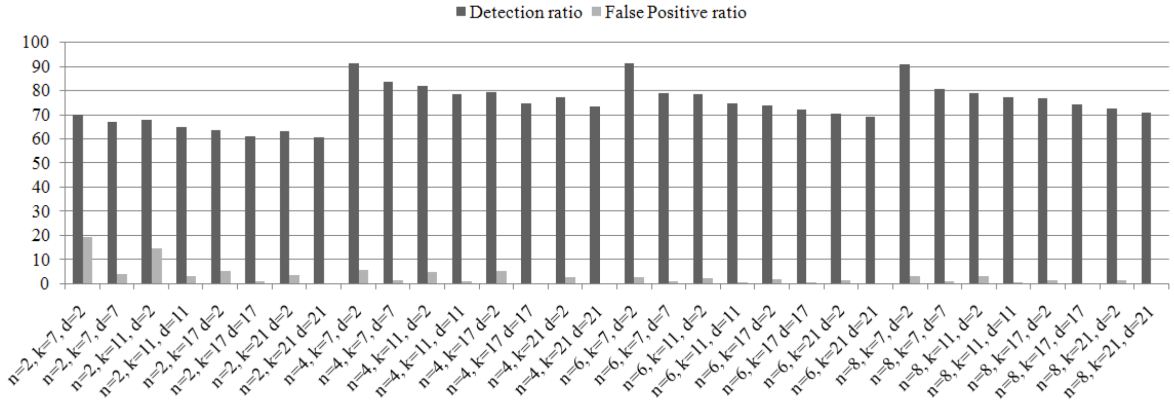

Figure 1: Detection and false positive ratio

in beforehand. It fails, however, when confronted to unknown malware.

In this way, our method uses n-grams signatures to achieve detection of unknown malware. In our experiment we have chosen a set of decompiled malware code and texts that have result of the monitorizing of the execution of those file to act as the unknown threats and other set to act as the known threats. Under these premises, we extract the n-grams of the known files and the n-grams of the unknown ones. For any instance of the unknown set, we are going to classify into malware or benign software, based upon the coincidences of n-grams between the set of n-grams of the known files and the set of n-grams of the unknown ones.

We have demonstrated that n-grams-based methodology signatures can achieve detection of new or unknown malware. Furthermore, with the inclusion of the parameter d in the classification method, we have achieved a way to configure how strict the system will be for classifying an unknown instance as malware; thus, it is a way for controlling the false positive ratio.

This method provides a good detection ratio and the possibility to control the false positive ratio. This way, this method can be applied in order to detect as much malware as it can or it can be configured to give a 0% false positive ratio.

Future work will focus on further research of the capability of n-gram analysis in malware detection, experiments with different and larger malware collections, and combination of this technique with behavioural analysis of malicious code.

REFERENCES

Abou-Assaleh, T., Cercone, N., Keselj, V., and Sweidan, R. (2004). N-gram-based detection of new malicious code. In *COMPSAC Workshops*, pages 41–42.

Fix, E. and Hodges, J. L. (1952). Discriminatory analysis: Nonparametric discrimination: Small sample performance. *Technical Report Project 21-49-004, Report Number 11.*

Jacob, A. and Gokhale, M. (2007). Language classification using n-grams accelerated by fpga-based bloom filters. In *HPRCTA '07: Proceedings of the 1st international workshop on High-performance reconfigurable computing technology and applications*, pages 31–37, New York, NY, USA. ACM.

Kaspersky (2008). Kaspersky security bulletin 2008: Malware evolution january - june 2008.

Kephart, J. O. (1994). A biologically inspired immune system for computers. In *In Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 130–139. MIT Press.

Morley, P. (2001). Processing virus collections. In *Proceedings of the 2001 Virus Bulletin Conference (VB2001)*, pages 129–134. Virus Bulletin.

Nachenberg, C. (1997). Computer virus-antivirus coevolution. *Commun. ACM*, 40(1):46–51.

Schultz, M. G., Eskin, E., Zadok, E., and Stolfo, S. J. (2001). Data mining methods for detection of new malicious executables. In *SP '01: Proceedings of the 2001 IEEE Symposium on Security and Privacy*, page 38, Washington, DC, USA. IEEE Computer Society.

Zhou, S. and Guan, J. (2002). Chinese documents classification based on n-grams. In *CICLing '02: Proceedings of the Third International Conference on Computational Linguistics and Intelligent Text Processing*, pages 405–414, London, UK. Springer-Verlag.