Application of Data Mining based Malicious Code Detection Techniques for Detecting new Spyware

Cumhur Doruk Bozagac

Bilkent University, Computer Science and Engineering Department,
06532 Ankara, Turkey
dorukb@cs.bilkent.edu.tr

Abstract. Over the past few years, a new computer security problem has arisen, Spyware. Anti-Virus techniques cannot deal with spyware, due to their silent infection techniques and their differences from a regular virus. Various Anti-Spyware programs have been implemented as a counter-measure but most of these programs work using the signature method, which is weak against new spyware. Schultz[1] has presented a data-mining framework that detects new, previously unseen malicious executables. However his paper was published in 2001 and there was no spyware in their training or test dataset. This paper takes Schultz[1]'s work as a candidate and applies its techniques against a new spyware dataset collected in 2005, to see whether their techniques can be used against this new threat.

1 Introduction

Spyware is any software installed on a computer without the user's knowledge, that gathers information about that user for later retrieval by whomever controls it. There are two types of spyware: malware and adware. Malware is any program that gathers personal information from the user's PC. Key loggers, screen capture devices, and Trojans are in this category. Adware is a program designed for showing a user advertisements, like homepage hijackers, pop-up windows and search page hijackers.

Spyware poses several risks. The most conspicuous is compromising a user's privacy by transmitting information about that user's behavior. However, spyware can also detract from the usability and stability of a user's computing environment, and it has the potential to introduce new security vulnerabilities to the infected host. Because spyware is widespread, such vulnerabilities would put millions of computers at risk.

Spyware programs have a series of traits that makes them difficult to detect and enables them to install themselves on numerous PCs for long periods of time[2]:

- They use almost perfect camouflage systems. They are normally installed on computers along with another kind of application: a P2P client, a hard disk utility…
- File names don't normally give any clues as to the real nature of the file, and so they often go unnoticed along with other legitimate application files.
- As they are not viruses, and they don't use a routine that can be associated with them, antivirus programs don't detect them unless, they are designed specifically to do so.

Panda Anti-Virus program has recently released a new online virus scanner, which can also detect spyware. According to their reports[2], in the first 24 hours of the operation, 84 percent of malicious code detected was spyware and the first 74 most detected malicious code were all spy programs.

Spyware can be detected through the effects it has on systems: use of system resources resulting in a slowdown of the PC; high level of Internet connection bandwidth consumption; complete loss of the connection or general system instability. They also often change settings of certain applications, such as the browser home page, or insert icons on the desktop. The main consequence of spyware on PCs is the gathering of information, including confidential details, making users feel uneasy about what is happening in their computer behind their backs.

Current virus scanner technology has two parts[1]: a signature-based detector and a heuristic classifier that detects new viruses. The classic signature-based detection algorithm relies on signatures (unique telltale strings) of known malicious executables to generate detection models. Signature-based methods create a unique tag for each malicious program so that future examples of it can be correctly classified with a small error rate. These methods do not generalize well to detect new malicious binaries because they are created to give a false positive rate as close to zero as possible. Whenever a detection method generalizes to new instances, the tradeoff is for a higher false positive rate. Heuristic classifiers are generated by a group of virus experts to detect new malicious programs. This kind of analysis can be time-consuming and oftentimes still fail to detect new malicious executables.

2 Methodology

Schultz[1] proposes using data mining methods for detecting new malicious executables. They apply three different algorithms with each one having its own feature extraction technique. The first one is RIPPER algorithm, which they only applied on Portable Executable (PE) format data using the Portable Executable header information extraction technique, so I skip this algorithm. The second algorithm is Naïve Bayes algorithm using strings in the binaries as features. This technique can be easily avoided by encoding or encrypting the strings in a file, so this technique is weak against new malicious code. The third algorithm is Multi-Naïve Bayes algorithm, which uses byte sequences in a file as features. The idea is, malicious executables have common intentions and they may have similar byte code. We can detect malicious executable by looking at the frequency analysis of byte code in a file. Multi-Naïve Bayes algorithm is basically a collection of Naïve Bayes algorithms for splitting the data into sets. I will use the same feature extraction technique, but I will use only one instance of Naïve Bayes algorithm. Source code for this method can be found at *http://www.cs.columbia.edu/ids/mef/software* .

The dataset consists of 312 benign(non-malicious) executables and 614 spyware executable. The spyware collection is formed by using the virus collection at *http://vx.netlux.org* and by crawling the internet using a sandboxed operating system and manually collecting spywares. The benign executables are collected from the system files in Windows XP operating system and from programs of a stereotype user.

Byte sequences are extracted using the *hexdump* tool in Linux. For each file in the dataset, using this tool a hexdump file is formed. When the algorithm is run, user should specify a "window size". Naïve Bayes algorithm can be specified to use how many sequences of byte data. Default window size is one, which means algorithm will look at 4 hexadecimals(2 bytes of data) for frequency analysis. I run the algorithm for a window size of 2 and 4 separately using 5-fold cross validation technique.

3 Test Results

To evaluate our system we are interested in several
Quantities, just like Schultz[1] :

1. True Positives (TP), the number of malicious executable examples classified as malicious executables

2. True Negatives (TN), the number of benign programs classified as benign.
3. False Positives (FP), the number of benign programs classified as malicious executables
4. False Negatives (FN), the number of malicious executables classified as benign binaries.

The "detection rate" of the classifier is the percentage of the total malicious programs labeled malicious. The "false positive rate" is the percentage of benign programs which were labeled as malicious. The "overall accuracy" is the percentage of true classifications over all data.

The first test is done using a window size of 2 and its results curve is shown in **Table 1.**

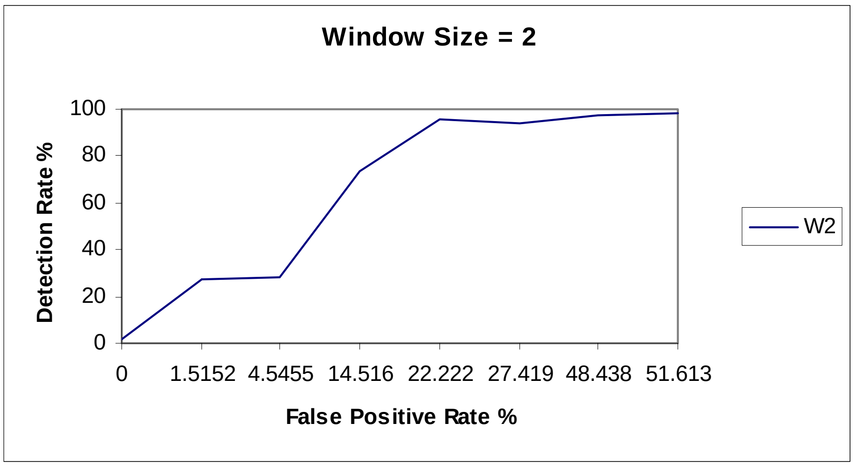

Table 1. Results of the tests for Spyware dataset.

As you can see, high detection rate can only be gained for high false positive rates and till 20% false positive rate the model has a detection rate lower than 80%. Then I run the algorithm with a window size of 4 and the results can be seen in **Table 2.**

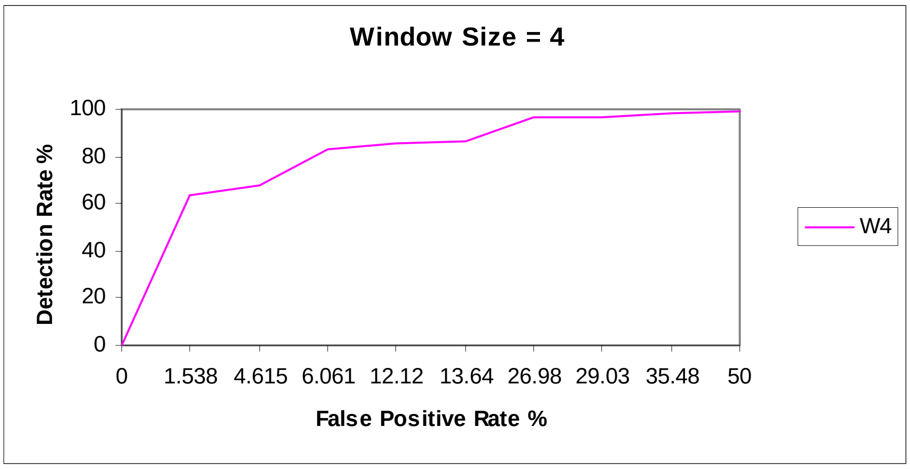

Table 2. Results of the tests for Spyware dataset.

Results were better for this case, but still the overall accuracy was rather low, so I decided to investigate the reasons by looking at the classification test results. I realized that a class of spyware called Trojans had a quite low detection rate. These are different from ordinary spyware, since they are quite complicated programs and their binary size were rather big compared to other files in the dataset. I thought that these programs may be the reason for the high false positive rate and low detection rate, so I run another test for a window size of 2, after excluding the 59 Trojans from the dataset.

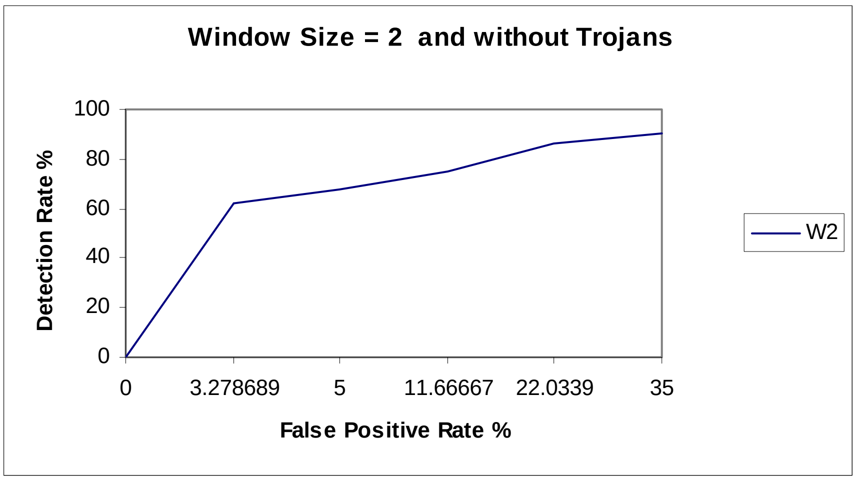

Table 3. Results of the tests for Spyware dataset.

As the **Table 3** shows, the overall accuracy has improved for the window size of 2 and we reach 80% detection rate before a false positive rate of 15%. Also we score better for low false positive rates. These results encouraged for a window size of 4 test without the Trojans in the dataset and the results of this test are shown in **Table 4.**

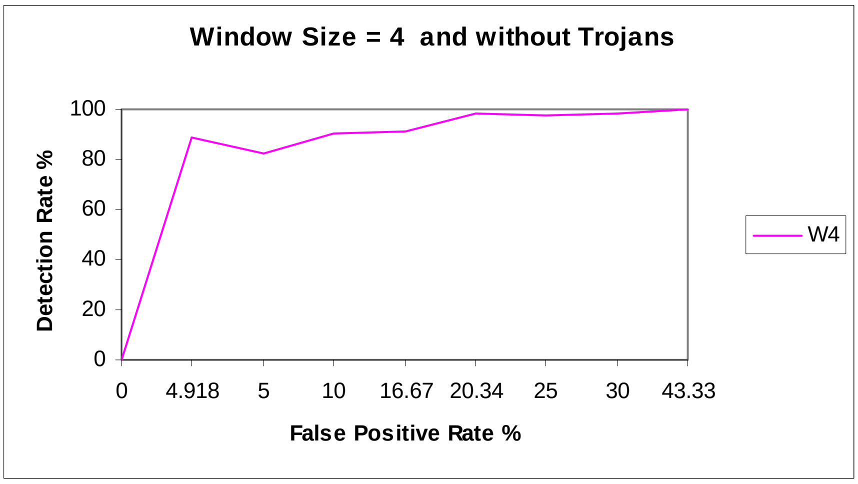

Table 4. Results of the tests for Spyware dataset.

We had 80% detection rate even before the false positive rate of 4% and overall accuracy has been improved. Below you can see best overall accuracy results for each run in **Table 5.**

	TP	TN	FP	FN	Detection Rate	False Positive Rate	Overall Accuracy
Window Size=2	118	49	14	5	95.93	22.22	89.78
Window Size=4	119	46	17	4	96.75	26.98	88.71
Window Size=2 w/o Trojan	96	46	13	15	86.49	22.03	83.53
Window Size=4 w/o Trojan	99	58	3	12	89.19	**4.92**	**91.28**

Table 5. Results of the tests for Spyware dataset.

The final run has the best overall accuracy and has the lowest false positive rate, which is significantly lower than other three runs.

4 Conclusion

Spyware is a new threat and current anti-virus programs are weak against spyware, because they use almost perfect camouflage systems and as they are not viruses, and they don't use a routine that can be associated with them, antivirus programs don't detect them unless, they are designed specifically to do so. There are some anti-spyware programs, but they work with signature scheme, so again they are weak against unseen spywares.

Before the tests, I had some doubt for using byte sequences as a feature. All viruses have a common technique for infection, which is replicating themselves onto new executables. One can argue that, the byte codes for infection can ve a distinguishing feature for the viruses and these can be used for detecting viruses. However, spyware does not use these techniques, so it is difficult to separate them from normal executables in this sense. On the other side, there is a common behaviour for all spyware, they watch the user's actions for reporting. Probably, this is why I get overall accuracy when I run tests with trojans in the dataset. Trojans are not really spyware, tough they are used for spying, indeed they are complete programs working as a toolbox for hackers. As a result, after removing them I had a dataset with a common programmatic behavour, which gives me better classifying results.

Another thing important about this technique is the window size. In the paper Schultz[1] does not give an optimal window size, but in general we can say that the larger window size we use, better results we get. The problem is, with a window size of five the program needs more than one gigabyte of ram for running and the run takes too much time and this is why I was only able to test with a window size of four.

To conclude, applying Schultz[1]'s techniques on spyware, I was able to see that a data mining based heuristic scheme has the potential to be used for detecting new spyware.

5 Future Work

Future work involves using Multi-Naïve Bayes for using a window size of five and six. I expect to have better overall accuracy for larger window sizes. Furthermore Christodorescu[3] proposes a technique for testing malicious code detector's performance against obfuscation. Since we are using byte sequences as our features, changes in the byte sequence will most probably affect our results. These techniques can be used to test performance of this spyware detection technique against obfuscated code.

6 References

1. M. G. Schultz, E. Eskin, E. Zadok, and S. J. Stolfo, **Data Mining Methods** for **Detection** of **New Malicious Executables**, Proceedings of IEEE Symposium on Security and Privacy. Oakland, pp.38-49, 2001

2. "The silent epidemic of 2005: 84% of malware on computers worldwide is spyware". http://www.pandasoftware.com/about/press/viewnews.aspx?noticia=5968.

3. M. Christodorescu and S. Jha, **Testing Malware Detectors**, International Symposium on Software Testing and Analysis archive. Boston, Massachusetts, USA.