Feature Set Selection in Data Mining Techniques for Unknown Virus Detection – A Comparison Study

Jianyong Dai
University of Central Florida
4000 Central Florida Blvd
Orlando, Florida, 32816
(407)823-2524
daijy@cs.ucf.edu

Ratan Guha
University of Central Florida
4000 Central Florida Blvd
Orlando, Florida, 32816
(407)823-2956
guha@eecs.ucf.edu

Joohan Lee
Tradeworx Inc
54 Broad Street, Suite 200
Red Bank, NJ 07701
(732)915-6139
joohan@tradeworx.com

ABSTRACT
Detecting unknown viruses is a challenging research topic. Data mining approaches have been used to detect unknown viruses. The key to data mining lies on the feature set to be used. There are several different approaches have been tried before, simple heuristics, static features and dynamic features. In this paper, we present several different data mining approaches and compare the result of these approaches.

Categories and Subject Descriptors
K.6.5 [**Computing Milieux**]: Security and Protection - Invasive software

General Terms
Security

Keywords
Data Mining, Virus Detection, Feature Set, Feature Selection

1. INTRODUCTION
Malicious software imposes a greater challenge to the computer researchers. The prevailing approach to detect virus is signature match. Anti-virus software comes with a signature database and signature database is updated constantly to be able to detect new viruses. While this approach works fine in old days, it is no longer effective in many cases. The advent of polymorphism and metamorphism makes signature based approaches ineffective. The propagation speed is much faster thanks to the era of internet and the result is that the signature database cannot keep up with.

To solve this problem, approaches to detect unknown viruses without the signature database have been proposed. Heuristics is widely used in commercial anti-virus software. For simple viruses, heuristics is effective. However, for advanced viruses, heuristics approach is too simple and could be easily defeated.

CSIIRW '09, April 13-15, Oak Ridge, Tennessee, USA

Another approach is data mining. The key component for a data mining approach is the feature set it is using. We classify existing data mining approaches into three categories. They are simple heuristic features, static features and dynamic features.

Simple heuristic features [1] are the features extracted from Win32 executable PE head or strings extracted from executable body.

N-Gram is consecutive bytes inside a binary code. [2, 3] derive feature set from N-Gram.

Static features are extracted through static analysis. Most approaches are based on assembly code of the unknown binary [4].

Dynamic features are features which could only be collected at runtime. Most dynamic approaches are based on dynamic system calls. However, we use dynamic instruction sequences in our experiments.

In this paper, we first present our approaches based on simple heuristics, N-Gram features, static instruction sequences features and dynamic instruction sequences features. Then we present our experimental result to compare all these approaches.

2. SIMPLE HEURISTICS
Simple heuristic features refer to the feature set extracted from Win32 PE head or strings inside executable body.

[1] illustrates many heuristics to detect structural anomaly of Win32 viruses. In our experiment, we use the following features:

1) Entry point locates in the last section
2) Suspicious section name
3) Suspicious section characteristics
4) Inconsistent size calculation
5) Suspicious imports by ordinal
6) Critical function imports
7) Corrupt relocation table
8) Occupation rate of last block in sections

All these features are binary and we use these 8 features to train and evaluate our data mining models.

3. N-GRAM APPROACH

Several approaches have been proposed to use N-Gram [2, 3]. N-Gram refers to consecutive binary sequences of fixed size inside binary code. We extract most frequent N-Grams from the training dataset which contains both benign software and malicious software, and use the occurrence rate of top N-Gram as our features.

4. STATIC INSTRUCTION SEQUENCES APPROACH

Static approaches are based on the features derived from static analysis. Some approaches are based on assembly code derived by disassembling the executable binary [4]. Other approaches are based on higher level structure based on assembly code such as control flow graph (CFG) [5]. However, disassembling binary code itself is a hard problem and no general solution has been discovered yet [6].

Here we propose to use opcodes derived from disassembly code as the criteria to derive our feature set. Our approach does not depend on the disassemblility and the disassembility becomes a heuristics in our approach.

We first disassemble the binary code into assembly code. The assembly code breaks into natural blocks. The start point of each block is associated with a disassembler-generated label. Usually it is a destination of a control flow instruction elsewhere in the program. For each block, we generate a line with a sequence of instruction operation codes inside the block as shown in Figure 1.

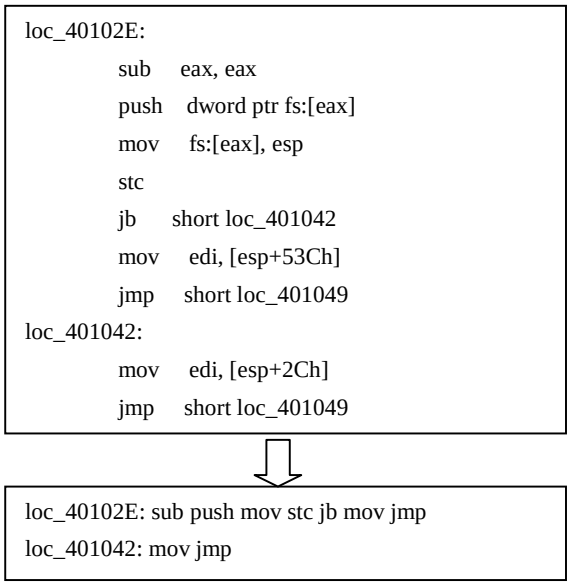

Figure 1 Abstract Assembly

If an executable cannot be fully disassembled, we extract fewer sequences or no sequences at all. Our process will still work even in these cases.

We pay attention to the opcode and ignore the operands and prefix since the opcode represents the behavior of the program. The resulting assembly code is called abstract assembly

Once we get abstract assembly, we are interested in finding relationship among instructions within each basic block. The instruction sequences we are interested in are not limited to consecutive and ordered sequences. Virus writers frequently change the order of instructions and insert irrelevant instructions manually to create a new virus variation. Further, metamorphic viruses automatically change the order of sequences and insert garbage among useful instructions to make new virus copies. The resulting virus variation still exhibits the same malicious behavior. However, any detection mechanism based on consecutive and ordered sequences such as N-Gram could be fooled.

We have two considerations to obtain the relationship among instructions. First, whether the order of instructions matters or not; Second, whether the instructions should be consecutive or not. Based on these two criteria, we use three methods to collect features.

1) The order of the sequences is not considered and there could be instructions in between.
2) The order of instructions is considered, it becomes instruction episodes. However, it is not necessary for instruction episodes to be consecutive.
3) The instructions are both ordered and consecutive.

We call these “Type 1”, “Type 2” and “Type 3” instruction associations.

The features for our classifier are instruction associations. To select appropriate instruction associations, we use the following two criteria:

1) The instruction associations should be frequent in the training dataset consisting of both benign and malicious executables.
2) The instruction associations should be abundant in benign code and rare in malicious code, or vice versa.

To satisfy 1, we use Apriori algorithm [7] to find frequent instruction associations only. One parameter of Apriori algorithm is “minimum support”. It is the minimal frequency of frequent associations among all data. More specifically, it is the minimum percentage of basic blocks that contains the instruction sequences in our case.

To satisfy 2, we define contrast as our criteria.

$$Contrast\ (Fi) = \begin{cases} \dfrac{Bcount\ (Fi) + \varepsilon}{Mcount\ (Fi) + \varepsilon} & Bcount\ (Fi) \geq Mcount\ (Fi) \\ \dfrac{Mcount\ (Fi) + \varepsilon}{Bcount\ (Fi) + \varepsilon} & Bcount\ (Fi) < Mcount\ (Fi) \end{cases}$$

BCount (F_i) normalized count of F_i in benign instruction file

MCount (F_i) normalized count of F_i in malicious instruction file

ε a small constant to avoid error when the denominator is 0

Only top contrast and frequent instruction associations will be retained as our feature set.

We then use the frequency of the selected instruction associations as our features and use these features to train and evaluate our data mining models.

5. DYNAMIC INSTRUCTION SEQUENCES APPROACH

Dynamic features refer to the features can be derived only by running the binary code. Most existing approaches are based on system calls [8, 9]. However, it takes time to collect runtime system call invocations. Instruction sequences are much faster to collect. In our experiment, we use an instruction sequences based approach to collect our dynamic feature set [10].

We capture instruction sequences at runtime and we get a chronicle record of all the instruction executed. Unlike assembly code in which instructions breaks naturally into blocks, we do not have such blocks in the execution log. To get assembly code similar to those we get in static analysis, we have to further process it to organize instructions into blocks.

The algorithm to organize instruction sequences into blocks consists of three steps:

1) Sort all instructions in the execution log on their virtual addresses. Repeated code fragments will be ignored.

2) Scan all jump instructions. If it is a control flow transfer instruction (conditional or unconditional), we mark it as the beginning of a new basic block.

3) Output sorted instructions and labels if applicable

Once we get the instruction sequence blocks, we will use similar process as in our static approach to get top instruction associations and derive our feature set.

6. EXPERIMENT RESULTS

We collect 267 Win32 viruses from VX heaven [11] and randomly choose 368 benign executables which consist of Windows system executables, commercial executables and open source executables. These executables have similar average size and variation as the malicious dataset.

For static and dynamic instruction sequence approaches, we try different setting of support level, instruction association type, support level and number of feature selected. For dynamic approach, we also try different number of instruction sequence captured. We choose the best model among all different settings for both static instruction sequence and dynamic instruction sequence cases. All approaches use two classifiers: C4.5 decision tree [12] and support vector machine [13]. We also tested some other classifiers, however, we do not detect any classifier has clear advantage over others in the measure of accuracy.

We use 5-fold cross-validation to evaluate the performance in our experiments. The major criteria we use to compare different approaches are testing accuracy. We also calculate and compare false positive rate and false negative rate.

Table 1 illustrates all results use C4.5 decision tree and table 2 illustrates all results use support vector machine. In each cell, there are two numbers representing the correspondent value over training dataset / testing dataset.

	Accuracy	FPR	FNR
PE consistency	90.0%/89.7%	2.6%/23.4%	2.8%/24.8%
8-Gram	98.6%/88.2%	0.9%/2.1%	11.1%/12.7%
Static IS	89.5%/86.3%	17.0%/20.6%	3.6%/6.6%
Dynamic IS	93.7%/91.0%	9.4%/12.6%	4.1%/6.5%

Table 1 C4.5 decision Tree results

	Accuracy	FPR	FNR
PE consistency	90.6%/85.3%	2.2%/22.5%	1.7%/23.3%
8-Gram	94.5%/89.3%	2.0%/10.2%	6.0%/17.2%
Static IS	94.3%/84.9%	4.1%/12.7%	8.3%/18.7%
Dynamic IS	95.4%/91.9%	7.4%/9.6%	2.5%/6.8%

Table 2 Support Vector Machine results

The main criteria to evaluate a model are accuracy over testing dataset. However, we also calculate false positive rate and false negative rate. False positive rate is the proportion of benign executables that were erroneously reported as being malicious. On contrary, false negative rate is the proportion of malicious executables that were erroneously identified as benign. We believe in a virus detection mechanism, low false negative rate is more vital than low false positive rate. It is wise to be more cautious against those suspicious unknown executables. High false positive certainly make things inconvenient for the user, but high false negative will destroy a user's computer, which is more harmful.

The detection rate for any approach is above 85%, which means all these approaches are effective to detect unknown viruses. In our result, dynamic approach achieves the best in terms of both accuracy and false negative rate. The best detection rate 91.9% is achieved by support vector machine using dynamic instruction features. And the false negative rate of this model is 6.8%, which is pretty low among all models.

The result between different classifiers is mixed. That mean no classifier is clearly superior to other classifiers. Dynamic instruction sequences are much better than other feature sets. However, the performance of other feature sets is quite mixed.

7. CONCLUSION

In this paper, we experiment four different approaches in detecting unknown computer viruses. The differences among all approaches are the feature set we are using. We use simple heuristic features, N-Gram features, static instruction sequences features and dynamic instruction sequences features. With this limited scale experiment, we find that dynamic instruction sequences features achieve slightly better result than other feature set we have tried.

8. REFERENCES

[1] Peter Szor, "The Art of Computer Virus Research and Defense", Addison Wesley, 2005.

[2] Tony Abou-Assaleh, Nick Cercone, Vlado Keselj, and Ray Sweidan, "Detection of New Malicious Code Using N-grams Signatures" in Proceedings of the Second Annual Conference on Privacy, Security and Trust (PST'04), Oct. 2004, pp. 193-196.

[3] Kolter, J.Z., and Maloof, M.A, “Learning to detect malicious executables in the wild” in Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2004, pp. 470–478.

[4] Muazzam Ahmed Siddiqui, "Data Mining Methods for Malware Detection," University of Central Florida, Tech. Rep., 2008, phD Dissertation.

[5] Mihai Christodorescu, Somesh Jha, “Static Analysis of Executables to Detect Malicious Patterns,” in 12th USENIX Security Symposium, 2003.

[6] Christopher Kruegel et al, "Static Disassembly of Obfuscated Binaries," in Proceedings of the 13th conference on USENIX Security Symposium, 2004.

[7] Rakesh Agrawal, Ramakrishnan Srikant, “Fast Algorithms for Mining Association Rules,” in Proc. 20th Int. Conf. Very Large Data Bases (VLDB), 1994.

[8] Steven A. Hofmeyr et al, "Intrusion detection using sequences of system calls," Journal of Computer Security, vol. 6, no. 3, 1998.

[9] Wenke Lee and Salvatore J. Stolfo, “Data Mining Approaches for Intrusion Detection,” in 7th USENIX Security Symposium, 1998.

[10] Jianyong Dai, Ratan guha, Joohan Lee, “Efficient Virus Detection Using Dynamic Instruction Sequences “, to appear in Journal of Computers, Volume 4, Issue 5, May 2009

[11] http://vx.netlux.org

[12] J.R.Quinlan, "C4.5:Programs for Machine Learning", Morgan Kaufmann, 1993.

[13] John Shawe-Taylor and Nello Cristianini, "Support Vector Machines", Cambridge University Press, 2000.