Intruder Detection Using Data Mining Techniques

A Srinivas[1], DR. Rama Raju P.hD [2]
Department of Computer Science and Engineering
C.M.R COLLEGE OF ENGINEERING
JNTU, Hyderabad, India.
sress916@gmail.com
Dr.Ramaraju65@gmail.com

ABSTRACT: **Anti-virus systems traditionally use signatures to detect malicious executables, but signatures are over fitted features that are of little use in machine learning. Other methods seek to utilize more general features, with some degree of success. Through this project, we presented a data mining approach that conducts an exhaustive feature search on a set of computer viruses. Data mining methods detect patterns in large amounts of data, and use these patterns to detect future instances in similar data. We can also use classifiers to detect malicious executables. A classifier is a rule set, or detection model, generated by the data mining approach that was trained over a given set of training data.**

Keywords: **Malicious Executable, Signature, Data Mining, Classifier.**

1. INTRODUCTION

There are many approaches used for detecting malicious program. But every year thousands of new viruses are found for that traditional approaches are not sufficient to detect those files. To address this problem, we explore solutions based on machine learning and not strictly dependent on certain viruses. 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.

Malicious software is becoming a major threat to the computer world. The general availability of the malicious software programming skill and malicious code authoring tools makes it easier to build new malicious codes. Recent statistics from Windows Malicious Software Removal Tool (MSRT) by Microsoft shows that about 0.46% of computers are infected by one or more malicious codes and this number is keep increasing [1].

Moreover, the advent of more sophisticated virus writing techniques such as polymorphism [2] and metamorphism [3] makes it even harder to detect a virus. The data-mining framework automatically found patterns in our data set and used these patterns to detect a set of new malicious binaries [4].

Our aim is to develop a more systematic and efficient approach in building virus detection model. In first section

Method we present whole model for select top L feature from malicious data set. We generate a data set of malicious programs and disassemble all files.

2. MALWARE TAXONOMY

Malware is a piece of code which changes the behavior of security sensitive applications, without a user consent and in such a way that it is then impossible to detect those changes using a documented features of the application. Fig. 2 shows the taxonomy of malware by william [9].

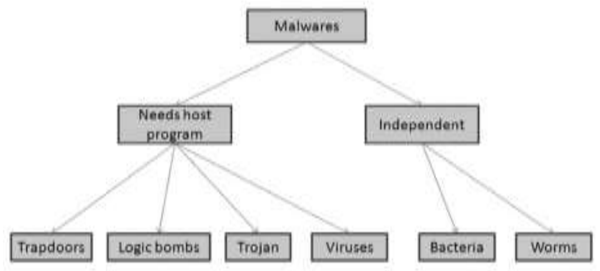
Trap Doors: It is a secret entry point into a program that allows someone that is aware of the back door to gain access without going through the usual security access procedures. This usually is done when the programmer is developing an application that has an authentic procedure, or a long setup, requiring the user to enter many different values to run the application.

Logic Bomb: A logic bomb is a piece of code intentionally inserted into a software system that will set of a malicious function when specified conditions are met. Logic bomb code is embedded in some legitimate program that is set to explode, when certain condition are met.

Trojan Horse: A Trojan horse is malware that appears to perform a desirable function for the user prior to run or install but instead facilitates unauthorized access of the user' s computer system. Trojans are hidden in programs which appear useful. We visit some site to download a program and then run the program now our system is infected.

Virus: A virus is a program that can infect application programs by modifying them. Modification includes a copy of the virus programs, which can infect other programs.

Worms: A computer worm is a self-replicating malware computer program. It uses a computer network to send copies of itself to other nodes (computers on the network) and it may do so without any user intervention.

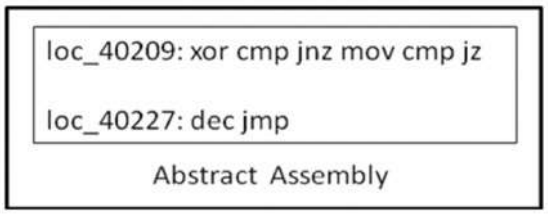


3. FEATURE DETECTI ON APPROACH FROM VIRUSES THROUGH MINING

M ethod

In this paper we present an virus detection approach through data mining. For that we used some virus files from corpus data set and some viruses generate from vcl32 virus kit.

First of all we take 2000 virus files from corpus data set and vcl 32 vi rus generator. Then through I Dpro disassembler, disassemble all virus file and generate ASM files from those. In a disassembler, assembly instructions are organized into basic blocks. We make logic assembly and abstract assembly from those files. Disassembler will generate a label for each basic block automatically. We believe that basic block capture the structure of instruction sequences and we process the instructions and make basic blocks. That code is " logic assembly" code [5]. Each assembly instruction consists of opcode and operands. We use only opcode and ignore the operands and prefix because that say behavior of program. The resulting assembly code is called "abstract assembly" [5]. Final abstract assembly as show below



loc_40209: xor cmp jnz mov cmp jz

loc_40227: dec jmp

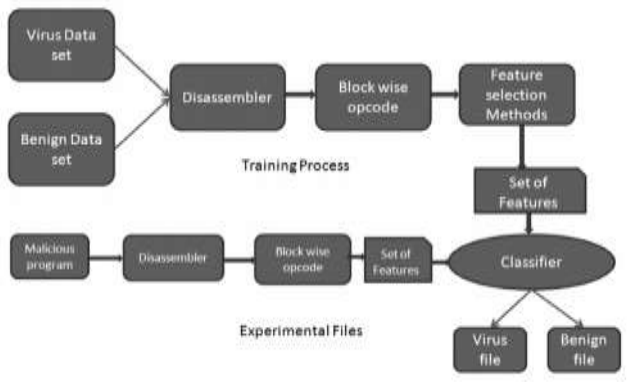
Abstract Assembly

Example of abstract assembly

Major steps

1. Make virus datasets.
2. Disassemble virus files using any disassembler.
3. Generate abstract assembly opcode.
4. Feature selection algorithm.

Step 4: Take ith no. of training files (virus and benign) and find frequency of each IA at block level.



4. FEATURE SELECTION

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 data set. If it occurs very rarely, we would rather consider this instruction association is a noise and not use it as our features.
2. The instruction associations should be an indicator of malicious code.

To satisfy the criteria, we only extract f requent instruction associations from training dataset. Only frequent instruction associations can be considered as our features. We use a variation of Apriori algorithm to generate all three types of frequent instruction associations from abstract assembly. One parameter of Apriori algorithm is "minimum support" [5]. 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. Normalized count is the frequency of that instruction sequence divided by the total number of basic blocks in abstract assembly. We can also use *N* gram approach to find feature set from that data [6]. Then select top *L* features as our feature set. For one executable in training dataset, we count the number of basic blocks containing the feature, normalized by the number of basic blocks of that executable. We process every executable in our training dataset, and eventually we generate the input for our classifier as like Naive Bayes, Ripper[8].

Following Steps are Shown Basic Architecture

Step 1: Disassemble all files and generate abstract assembly.

Step 2: Find frequency of each instruction association (IA) according Type 1 and 2

Step 3: Sort all instruction sequence and select top 10 sequences of length k.

Step 5: Make table of selected IA frequencies from training files.

Step 6: Repeat step 4 and 5 for Type 1, 2 and length 2, 3 IA.

Algorithm

Find the frequent itemsets: the sets of items that have minimum support

INPUT: Set of virus files (*V*)

OUTPUT: Set of top instruction sequences (*L*).

In order to generate set of instruction sequences we have set of virus file. In each virus file we have no. of basic blocks. Form the basic blocks occurrence of instruction sequences is calculated, which is called as instruction association. This algorithm repeats until all set of virus file encountered. Finally we select top *L* sequences which are called as top L virus features.

Following are the basic steps for generating top *L* instruction sequences.

```
1. For (each virus file Vi in V) do
2. For (each basic block Bij in Vi) do
3. Record all sequences of length sl found in Bij (with out
   repetition)
4. Increase count of all instruction sequences.
5. End For
6. End For
7. Select top L sequences.
```

5. EXPERIMENTAL SETUP

Virus data set:

(i) 1500 files from corpus data set [7]

(ii) 500 files from vcl32 generator

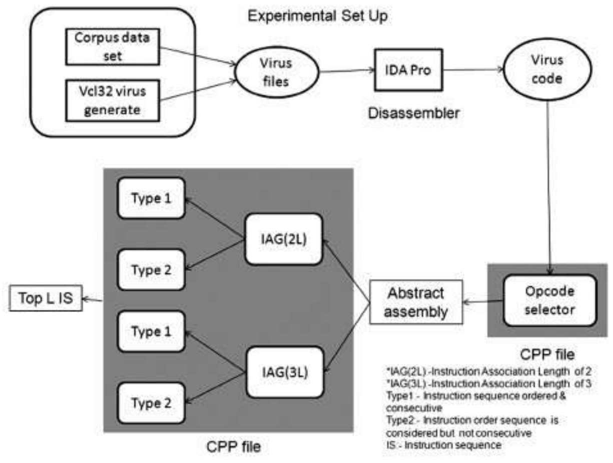
IDA Pro: Disassembler to generate ASM file from malicious files

Virus Code: ASM file of any virus file

Opcode selector: select opcode from asm files and make logic assembly and abstract assembly.

Abstract assembly: Opcode of all virus file as per basic blocks.

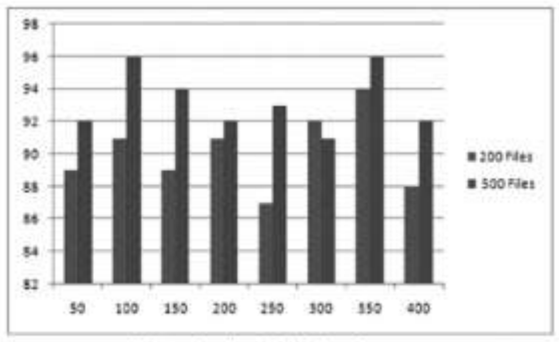
In above fig virus files are generated from VCL32 and corpus data set. Through Idpro disassembler we generate instruction code of those files. We present whole model for select top *L* feature from malicious data set. We generate a data set of malicious programs and disassemble all files. Then we use opcode selector for refine virus code and generate abstract assembly.


Experimental Set Up
Corpus data set
Vcl32 virus generate
Virus files
IDA Pro
Disassembler
Virus code
Type 1
Type 2
IAG(2L)
Type 1
Type 2
IAG(3L)
Top L IS
Abstract assembly
Opcode selector
CPP file
CPP file
*IAG(2L) -Instruction Association Length of 2
*IAG(3L) -Instruction Association Length of 3
Type1 - Instruction sequence ordered & consecutive
Type2 - Instruction order sequence is considered but not consecutive
IS - Instruction sequence

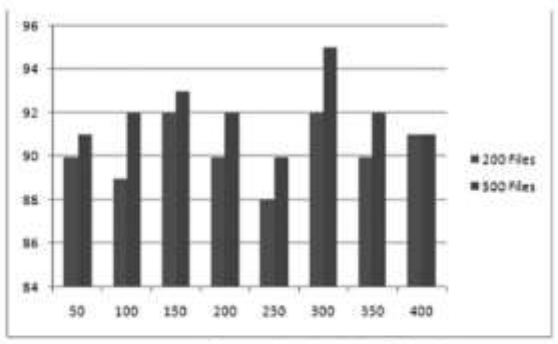
5.1. Results

(a) *Model Trained by Neural Network Classifier*

Following results are comparison between 500 files and 200 files trained by NN model. Graph shows better results for NN model which is trained by 500 files as compared to NN model trained by 200 files. From the help of Graph it is concluded that NN model trained by more files produces better results.



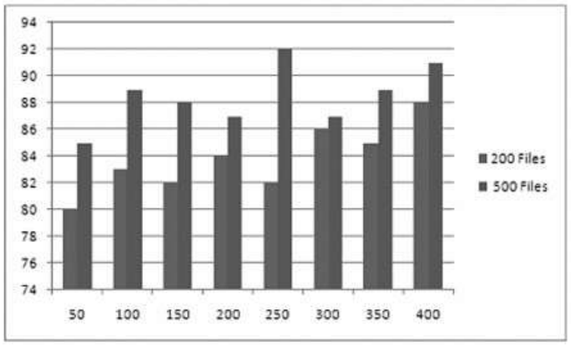
Instruction Association Type 1



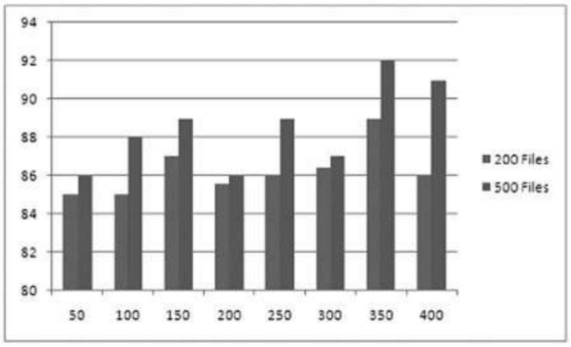
Instruction Association Type 2

(b) Model Trained by SVM Classifier

Following results are comparison between 500 files and 200 files trained by SVM model. Next graph shows better results for SVM model which is trained by 500 files as compared to SVM model trained by 200 files. From the help of graph it is concluded that SVM model trained by more files produces better results.



Instruction Association Type 1



Instruction Association Type 2

6. CONCLUSI ON

We implemented a feature search method that focuses on selecting features that are applicable to different families of viruses. This ensured that our classifier does not rely on signatures. In experimental testing our method achieved better performance as compared to some of older virus detection techniques. By using both SVM and NN models, the selected features which are used by the classifier produces overall support within the data set. This indicates that our feature search method produces features which are more useful while detecting new unseen viruses.

We also introduced an evaluation method for virus classifiers that tests more convincingly its ability to detect new viruses. Our results show that system which uses family non-specific features performs better results. In future work we propose focusing on reducing the false positive rate, by using a large number of benign files, or by training our classifier using a cost matrix and setting a higher cost to misclassifying negative examples. This would involve by using a set of older viruses in the training set and a set of more recent ones in the test set.

REFERENCES

[1] Microsoft Antimalware Team, " Microsoft Security Intelligence Report (January - June 2007)",

[2] C. Nachenberg, " Computer Virus-Antivirus coevolution",
Communications of the ACM, 40, No. 1.

[3] P. Szor and P. Ferrie, " Hunting for Metamorphic", *in 11th International Virus Bulletin Conference*, 2001.

[4] Data Mining Methods for Detection of New Malicious Proceedings of the 2001 IEEE Symposium on Security and Privacy Page: 38 Year of Publication: 2001 ISSN: 1081-6011

[5] " Efficient Virus Detection Using Dynamic Instruction Sequences Jianyong Dai, Ratan Guha", *Joohan Lee Journal Of Computers*, 4, No. 5, MAY 2009. University of Central Florida.

[6] A Feature Selection and Evaluation Scheme for Computer Virus. This Paper Appears in: Data Mining, 2006. ICDM '06. Sixth International Conference on Publication Date: 18-22 Dec. 2006 on page(s): 891-895, ISSN: 1550-4786,
ISBN: 0-7695-2701-7 INSPECAccession Number: 10222296 Digital Object Identifier:10.1109/ ICDM.2006. 4 Current Version Published: 2007-01-08.

[7] Vx heavens. http://vx.netlux.org/lib.

[8] A Data Mining Framework for Building Intrusion Detection Models. Wenke Lee; Stolfo, S.J.; Mok, K.W. Security and Privacy, 1999 . *Proceedings of the 1999 IEEE Symposium* on 9-12 May 1999 Page(s):120-132, Digital Object Identifier 10.1109/SECPRI. 1999.766909.

[9]. Network Security Essentials, by William Staling