DETECTING TROJAN HORSES BASED ON SYSTEM BEHAVIOR USING MACHINE LEARNING METHOD

YU-FENG LIU[1], LI-WEI ZHANG[2], JIAN LIANG[2], SHENG QU[3], ZHI-QIANG NI[3]

[1] Data Mining Group, Software School, Tsinghua University, Beijing, China
[2]MOST Information Center, Beijing, China
[3]Network Service Group of Guo He LTD., Beijing, China
E-MAIL: liuyufeng_hit@yahoo.com.cn, liangjian@most.cn, nessus@gmail.com

Abstract:

The Research of detection malware using machine learning method attracts much attention recent years. However, most of research focused on code analysis which is signature-based or analysis of system call sequence in Linux environment. Obviously, all methods have their strengths and weaknesses. In this paper, we concentrate on detection Trojan horse by operation system information in Windows environment using data mining technology. Our main content and contribution contains as follows: First, we collect Trojan horse samples in true network environment and classify them by scanner. Secondly, we collect operation system behavior features under infected and clean circumstances separately by WMI manager tools. And then, several classic classification algorithms are applied and a performance comparison is given. Feature selection methods are applied to those features and we get a feature order list which reflects the relevance order of Trojan horse activities and the system feature. We believe the instructive meaning of the list is significant. Finally, a feature combination method is applied and features belongs different groups are combined according their characteristic for high classification performance. Results of experiments demonstrate the feasibility of our assumption that detecting Trojan horses by system behavior information is feasible and affective.

Keywords:

Trojan horse; System behavior; Classification; Feature selection;

1. Introduction

Computer attacks grew at an alarming rate nowadays as the popularization of network. Modern malwares are more likely to be aimed at stealing data for economic motivation. So the major security threat, no matter individual or government organization, is not only coming from traditional malicious software, such as worm, virus, etc, but more and more from Trojan horses, bonnet, etc.

Trojan horses are computer programs that presented as useful or harmless in order to induce the user to install and run them, but also have some hidden malicious goal, such as enabling remote access and control with the aim of gaining full or partial access to the infected system. On the one hand, since polymorphism of Trojan horses, it is hard for the signature-based technology, which is the mainly used method in current anti-virus program, detecting them; On the other hand, Trojan horses have very little immediate impact on the normal operation of a system, and they may go under detected for a significant period of time, allowing the attacker a large window of opportunity. We try to find some discriminative pattern using system behavior information to detect Trojan horses, and then defend against harmful activity.

In the following section, we review related work in section 2. And then we introduce the experiment in detail and analysis the experiment result in section 3, 4. Finally, we conclude in section 5.

2. Related work

There have been many attempts to use machine learning and data mining for the purpose of detecting malware [1, 2, 3, 4, 5, 6]. However, these attempts have concentrated mostly on malware such as worms, viruses, etc. Few prior works has specially target on Trojan horse. What's more, they mainly use malwares themselves, the malicious code, as analytical data source to find patterns and rules. The limitation of such approaches lies in the following aspects. Firstly, as new malware appears, new rules should be added to virus scanners to detect them, for signature-based scanner system in particular. Secondly, some malware, such as Trojan horses, are elusive. They are hard to detect, then it is impossible to use scanner to scan their codes for finding them. Many attempts are tried on computer systems running LINUX, UNIX operating system, for which viruses pose little threat. Such efforts are of little direct use for computers running the Microsoft's Windows operating system, for which malware pose great threat. In this paper, we concentrate on the Trojan horses in Microsoft windows environment.

2.1. Malware and Trojan horses

Malwares are various. The Trojan horse is one of the most complicated kinds. A Trojan horse may combine multiple techniques and is quite difficult to be clear. Micha Moffie studied the set of the Trojan Horses detecting six distinct characteristics and behaviors in [7], such as executing without user intervention, and initiating a connection to a fixed remote host, etc. From an attacker's point of view, his malicious program - the Trojan horse - is operating in an unfriendly environment. First, a user cannot control the malicious code, nor can the attacker until a connection is made. This means that the malicious code must be self-contained, and it must execute without any guidance from the user or the attacker. Second, the user or an anti-virus program may try to terminate the program as soon as it is detected; it is therefore beneficial for the program to hide and disguises itself as well as conceals its execution. The common execution patterns of Trojan Horses are summarized as following: 1.The malicious code is executed without user intervention.2. The malicious code may be directed by the remote attacker once a connection is established.3. Resources used by the malicious code, such as file names and network addresses, are hard-coded in the binary.4. OS resources (Processes, memory) used by the malicious code may be consumed for the purpose of degrading performance.

As the characteristics and behaviors of the Trojan horses mentioned above, we consider to using system behavior information to detect Trojan horses. It means we assume that when a Trojan horse infected a system and then there will be some recognizable system characters can be used to detect it. For example, if a Trojan horse is trying to connect a host over and over again because it doesn't have legitimate right. This seldom happen in a normal behavior. Then it will reflect on system behavior information parameter such as network, thread and IP, etc.

2.2. System behavior information

In a windows system, there are many kinds of system information. In this paper, we collected system behavior information by Windows Management Instrumentation (WMI). WMI is the infrastructure for management data and operations on Windows-based operating systems. WMI can be used in all Windows-based applications, and is most useful in enterprise applications and administrative scripts. We used WMI scripts to get the primary real-time system information which is high relevant with the behavior Trojan horses.

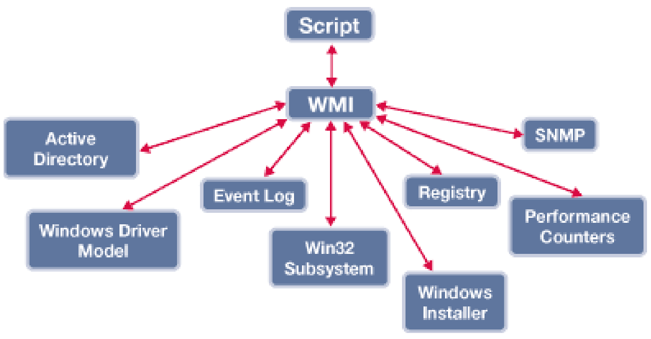

Figure 1.WMI model

2.3. Machine learning based methods

Our approach drew techniques machine learning mainly including classical classification methods and feature select technology, which will be introduced detailed below.

Instance based Learner (KNN)

One of the classical learning methods is the instance-based (IB) learner [8]. Its concept description is a collection of training examples or instances. Learning, therefore, is the addition of new examples to the collection. To classify an unknown instance, the performance element finds the example in the collection most similar to the unknown and returns the example's class label as its prediction for the unknown. For binary attributes, a convenient measure of similarity is the number of values two instances have in common. Variants of this method, such as KNN, find the k most similar instances and return the majority vote of their class labels as the prediction. Values for k are typically odd to prevent ties. Such methods are also known as nearest neighbor and k-nearest neighbors.

Naïve Bayes

Naïve Bayes is a probabilistic method that has a long history in information retrieval and text classification [9]. It stores as its concept description the prior probability of each class, *P(ci)*, and the conditional probability of each attribute value given the class, *P(vj |Ci)*. It estimates these quantities by counting in training data the frequency of occurrence of the classes and of the attribute values for each class. Then, assuming conditional independence of the attributes, it uses Bayes' rule to compute the posterior probability of each class given an unknown instance, returning as its prediction the class with the highest such value:

$$C = argmaxP(ci)\prod P(vj|ci)$$

Decision Trees

A decision tree is a tree with internal nodes corresponding to attributes and leaf nodes corresponding to class labels. For symbolic attributes, branches leading to children correspond to the attribute's values. The performance element uses the attributes and their values of an instance to traverse the tree from the root to a leaf. It predicts the class label of the leaf node. The learning element builds such a tree by selecting the attribute that best splits the training examples into their proper classes. It creates a node, branches, and children for the attribute and its values, removes the attribute from further consideration, and distributes the examples to the appropriate child node. This process repeats recursively until a node contains examples of the same class, at which point, it stores the class label. Most implementations use the gain ratio for attribute selection [9], a measure based on the information gain. In an effort to reduce overtraining, most implementations also prune induced decision trees by removing sub trees that are likely to perform poorly on test data [10].

Feature selection

There are various feature selection methods. The most wildly used including Document Frequency (DF), Information Gain (IG), Mutual Information (MI), and x^2 statistic (CHI), etc. In this paper, we choose the most extensively used method: IG method.

$$IG(j) - \sum_{vj\in\{0,1\}}\sum_{c\in\{Ci\}} P(vj,C)log\frac{P(vj,C)}{P(vj)P(C)}$$

where C is the class, *vj* is the value of the *jth* attribute, *P(vj, C)* is the proportion that the *jth* attribute has the value *vj* in the class *Ci, P(vj)* is the proportion that the *jth* n-gram takes the value *vj* in the training data, *P(C)* is the proportion of the training data belonging to the class C. This measure is also called average mutual information [11].

3. Experiment and evaluation

3.1. Data collection

Since no public standard dataset was available for this study, we had to create our own dataset. Firstly, we collected several Trojan horses which were representative. Then we created a computer environment which is a windows XP system running on VMware. We injected Trojan horses into the windows XP system separately maintain network is linked all the time. At the same time, we monitored various computer features and stored them with MySql database.

We used the MS Windows performance tool, which enables monitoring of system features that appear in 11 categories, which are stored in 11 tables. We found that 3 of the tables are useless in the analysis, so we drop them. Then there are 8 tables in our experimental dataset, as Table1. As shown in the table, there are so many attributes that we should try to check out which attributes really have relationship with the behavior of Trojan horses.

There are three kinds of Trojan horses. About 35000 records are collected in infected system, while about 15000 records are collected in clean system. Then we get dataset A which includes 8 tables, and every table has 50000 records and 2 classes. One class is labeled as bad, the other is benign. We get another dataset B using the same method, which including 1000 benign records, 2000 bad records which is generated by another kind Trojan horse. The dataset B also contains 8 tables.

The Trojan horses codes used in our experiments are collected from real network environment. Their names are test X version, total test, test high, test amazing version. We scanned them by VIRUSTOTAL (http://www.virustotal.com/) and found that they were the polymorphism of one category Trojan horse--Trojan.Win32.Agent. Their main behavior characters are as follows: when the users are unconscious, it will auto run and then download the other malwares to host PC from internet. At the same time, it will try to terminate the antivirus software.

TABLE 1 SYSTEM BEHAVIOR CATALOG

category	Attributes number	category	Attributes number
IP	17	processor	2
Network	16	process	15
TCP	9	system	17
thread	4	object	6

3.2. Experiment

We conducted three experimental studies using our data collection and experimental methodology, described previously.

In the first stage, we applied several learning methods to the dataset A to building classification model with all attributes. These learning methods are implemented in WEKA: Lazy.IBk (KNN), Trees.J48, and Bayes.NaiveBayes. In this phase, we get the classification precision for every table separately. And we used stratified ten-fold cross-validation. That is, we randomly partitioned the executables into ten disjoint sets of equal size, selected one as a testing set, and combined the remaining nine to form a training set. We conducted ten such runs using each partition as the testing set. The experiment result is shown in Table 2.

We found that KNN algorithm cost the longest time, while the precision of J48 is low relatively. Considering the efficiency and accuracy rating, we chose NaiveBayes as the most suitable classification method. We applied it to all tables in the second step. We used different test dataset. One is the training set itself using ten-fold cross-validation, the other is dataset B as mentioned hereinbefore, and we get the experiment result as Table 3 and Table 4.

The experiment showed that different test dataset had different accuracy. The most importance point was that the comparison reflected the limitation of data sample will affect the result. So the higher accuracy we want to get, the more samples we need that we used to collect the training data. What's more, as we compared the True Positive ratio (TP) and False Positive ratio (FP) of class B (which is defined as infected by Trojan horse), we found that attributes in tables: *objects, system, process* have better performance. It means they have good accuracy of classification on both classes, as showed below in Figure 2.

TABLE 2 PILOT EXPERIMENT WITH ALL ATTRIBUTES

Table	Learning method	Precision (A)	Precision (B)
Perfos_ objects	Lazy.IBk(KNN)	1	1
	Trees.J48	1	0.998
	NaiveBayes	1	0.998
Perfos_ processor	Lazy.IBk(KNN)	0.336	0.691
	Trees.J48	0.335	0.702
	NaiveBayes	0.341	0.765
Perfos_s ystem	Lazy.IBk(KNN)	0.344	1
	Trees.J48	0.331	0
	NaiveBayes	0.668	1

TABLE 3 NAIVEBAYES (TEN-FOLD CROSS-VALIDATION)

Table	Precision(A=0)	Precision(B)
Perfos_objects	1	0.999
Perfos_processor	0.653	0.972
Perfos_system	o.999	0.998
Perfproc_process	1	0.999
Perfproc_thread	0.569	0.848
Tcpip_ipv4	0.773	0.813
Tcpip_internetwork	0.765	0.806
Tcp_tcpv4	0.734	0.943

In the third stage, we use the Ranker and infoGain -AttributeEval in WEKA for attribute selection, if the threshold of the relevance is above 0.5, we choose it as high relevance attribute. If number of attributes whose threshold above 0.5 is less than 3 in a table, we select 3 from the top down according the IG rank. We get an attribute list as Table 5. Selection of parameters is according to experience and observational for getting better experiment performance. We expect more automatic method in future work.

At last, we try to combine attributes in different tables to get high precision. 8 tables were integrated to 4 tables. The test dataset was integrated the same. Then we get the experiments result as the Table 6. The principle of combination is the similarity of precision of classification.

TABLE 4 NAIVEBAYES (DATASET B AS TEST DATASET)

Table	Precision(A=0)	Precision(B)
Perfos_objects	1	0.998
Perfos_processor	0.337	0.759
Perfos_system	o.668	1
Perfproc_process	1	0.992
Perfproc_thread	0.361	0.708
Tcpip_ipv4	0.222	0.569
Tcpip_internetwork	0.293	0.655
Tcp_tcpv4	0.198	0.641

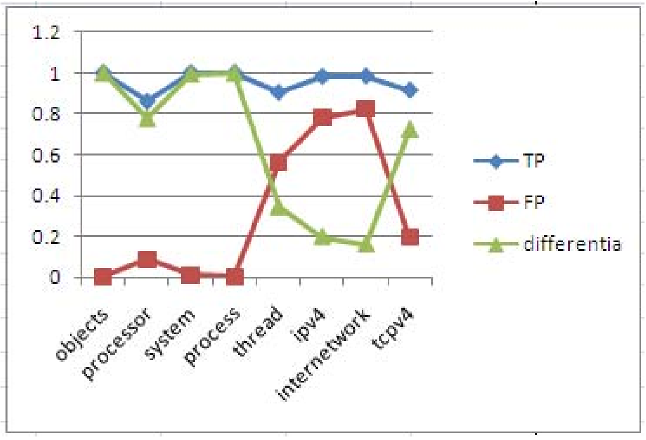

Figure 2 TP, FP in different tables

From the experiment result, it is easy to find that (1) different tables have different classification precision, and then we conclude that the attributes belonging to different system behavior catalogs make different contributions to classification. (2) The combination of attributions in different tables some increases the accuracy, such as system&tcpv4, and some get middle value of two tables. It showed that more proper attributes will get higher accuracy, while addition of improper attributes will decrease the accuracy instead.

4. Discussion and future work

The number of samples of Trojan horses collected for Experiments from real network environment in this paper is limited. So in the future work, we need to collected more samples to monitor system behaviors. Then we can ensure this method is working in real environment where all kinds of Trojan horses exist.

TABLE 5 IG OF ATTRIBUTE (THRESHOLD>0.5)

TABLE	IG	ATTRIBUTE NAME
Perfos_ objects	0.766	semaphores
	0.766	processes
	0.765	mutexs
	0.63	sections
	0.629	threads
	0.602	events
Perfos_ system	0.76608	systemmuptime
	0.76575	processes
	0.64	thread
Perfproc_ process	0.766	handlecount
	0.639	threadcount
Tcpip_ipv4	0.674336	datagramsreceiv edheadererrors
	0.622989	datagramsreceiv edaddresserrors
Tcpip_ tcpv4	0.765	connectionsreset
	0.621	connectionsestablishe

TABLE 6 ATTRIBUTES INTEGRATION

Table	Precision(A)	Precision(B)
Perfos_objects& Perfproc_process	1	0.998
Perfos_processor& Tcpip_internetwork	0.347	0.719
Perfos_system& Tcpip_tcpv4	0.751	0.999
Perfproc_thread &tcpip_ipv4	0.238	0.583

In real environment, there is certain to many programs are running in the same time with users' operation. However, when we collect system behaviors there is no any operation in our experiments. Because in this paper, we just want to validate that if the system behavior have some patterns that can detect Trojan horse. If we want apply this method to real environment, we should consider simulating the operation in real world in the future work.

The process we collect and preprocess the dataset is relative complicated. As we found few revenant attributes can almost represent all the attributes, we can try to use this character to simplify the process in the future work.

In addition, we can analysis the relationship between system behavior and the Trojan horses further according the result of feature selection.

5. Conclusions

We consider to applying the machine learning techniques to solve the problem of detecting Trojan horses in Microsoft windows environment. There indeed exist discriminate characters of data classification on them to detect if the system is infected by Trojan horses. After evaluating a variety of classification methods, experiment results show that NaïveBayes has more superiority on both efficiency and accuracy than other classical methods.

In this paper, we also show that the accuracy of classification may increase when the more relevant features are used in the process. However, the more features are selected, the more time building classification cost. It means that Trojan horses' detection system based on too many features will respond slowly in real time. Then, we conclude that the accuracy of classification the infected and un-infected system behavior mainly depending on the most important features, which can be selected using feature selection techniques. The Experiment shows that the accuracy of selected features is almost the same as whole features. This greatly reduces the consumption of time space. This may offer us another method for detecting Trojan horses in real environment.

We also compare the classification accuracy on same training dataset with different test dataset. The result shows the limitation of the samples, in the other word, the dependency of the training data in same extent.

References

[1] A. P. Kosoresow and S. A. Hofmeyr. Intrusion detection via system call traces. IEEE Softw., 14(5):35–42, 1997.

[2] AK Ghosh, C Michael, M Schatz. A real-time intrusion detection system based on learning program behavior.Lecture notes in computer science, 2000

[3] M. Schultz, E. Eskin, E. Zadok, and S. Stolfo. Data mining methods for detection of new malicious executables. In Proceedings of the IEEE Symposium on Security and Privacy, pages 38-49, Los Alamitos, CA, 2001. IEEE Press.

[4] Jeremy Z. Kolter , Marcus A. Maloof, Learning to detect malicious executables in the wild, Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, August 22-25, 2004, Seattle, WA, USA

[5] Mr. Dima Stopel. Detecting Malicious Code Activity Based on Computer Behavior Using Artificial Neural Networks.Thesis submitted in partial fulfillment of the requirements for the M.Sc.Ben-Gurion University of the Negev Faculty of Engineering Science Department of Information Systems Engineering, August, 2007.

[6] K Rieck, T Holz, C Willems, P Düssel, P Laskov. Learning and Classification of Malware Behavior Detection of Intrusions and Malware, and Vulnerability Assessment, 2008:108-125

[7] Micha Moffie, David Kaeli. Hunting Trojan horse. NUCAR, Technical Report TR-01, January 2006

[8] D. Aha, D. Kibler, and M. Albert. Instance-based learning algorithms. Machine Learning, 6:37-66, 1991.

[9] M. Maron and J. Kuhns. On relevance, probabilistic indexing and information retrieval. Journal of the ACM, 7:216-244, 1960.

[10] J. Quinlan. C4.5: Programs for machine learning. Morgan Kaufmann, San Francisco, CA, 1993.

[11] Y. Yang and J. Pederson. A comparative study on feature selection in text categorization. In Proceedings of the Fourteenth International Conference on Machine Learning, pages 412-420, Morgan Kaufman San Francisco, CA, 1997.