INTRUSION DETECTION BASED ON DATA MINING

Keywords: Intrusion detection, anomaly detection, user anomaly, data mining, frequent episodes

Abstract: Traditional computer misuse detection techniques can identify known attacks efficiently, but perform very poorly in other cases. Anomaly detection has the potential to detect unknown attacks, however, it is a very challenging task since it is aimed at the detection of unknown attacks without any priori knowledge about specific intrusions. This technology is still at its early stage. Existing research in this area focuses either on user activity (macro-level) or on program operation (micro-level) but not on both simultaneously. In this paper, an attempt to look at both concurrently is presented. Based on an intrusion detection framework (Lee, 2001), we implemented a user anomaly detection system and conducted several intrusion detection experiments by analysing macro-level and micro-level activities. User behaviour modelling is based on data mining; frequent episode algorithms are used to build the user's normal profiles. The experimental results have shown that the system can detect anomalies and changes in the user's normal working patterns effectively.

1 INTRODUCTION

In recent years, intrusion and other types of attacks to the computer network systems have become more and more widespread and sophisticated. In addition to intrusion prevention techniques such as authentication, firewall and cryptography, intrusion detection is often used as a second line of defence to protect computer networks.

Intrusion detection techniques can be classified into two categories: *misuse detection and anomaly detection* (Lee, 2001). Misuse detection looks for signatures of known attacks, and any matched activity is considered an attack. Misuse detection can detect known attacks efficiently, but it performs poorly with unknown attacks. Anomaly detection constructs models of subject behaviour, and any significant deviation from normal behaviours is considered part of an attack. Anomaly detection has the potential to detect unknown attacks since no advance knowledge about specific intrusions is required (Lee, 2001). In practice, both techniques are used in an intrusion detection system as they complement each other.

Accuracy, extensibility and adaptability are three important characteristics of intrusion detection systems (Lee, 2001). Since attacks change their types and patterns frequently, any intrusion detection system needs to update the detection rules dynamically so that it can notice new attacks. To remain efficient Intrusion detection systems should not use too much of system resources, as it would heavily affect the normal activities and would reduce overall efficiency.

A number of techniques such as data mining, statistics and genetic algorithms have been used for intrusion detection on the user and program activity levels individually. In (Lee 2001), a novel framework, called MADAM ID, for Mining Audit Data for Automated Models for Intrusion Detection is presented. It uses data mining algorithms to compute activity patterns from system audit data and extracts predictive features from the patterns. It then applies machine-learning algorithms to the audit records that are processed according to the feature definitions and generates intrusion detection rules. The data mining algorithms used to generate the patterns are meta-classification, association rules and frequent episode algorithms. The test results in 1998 conducted by DARPA Intrusion Detection Evaluation showed that the model was one of the best performing of all the participating systems in off-line mode. In order to detect user anomalies, the normal user activity profiles are created and a

similarity score range (upper and lower bound) is assigned to each user's normal pattern set. When in action, the system computes the *similarity score* of the current activity's patterns, and if this score is not in the *similarity score range*, the activity is considered as an anomaly.

Pikoulas et al (Pikoulas, 2001) have built an agent-based intrusion detection system using Bayesian forecasting technique to predict user action. The system was built on multi-agent techniques, in which a *core* agent sitting on a server keeps all user profiles, and the *user agents* are on the workstations. The user profile will be downloaded automatically from the *core agent* to *user agent* when the user starts a session. The system is reported to be able to determine user's unpredictable activities or changes in normal working patterns.

On the program activity level (micro-level), Warrender *et al* have pointed out that a number of machine-learning approaches such as rule induction, hidden Markov model can be used to learn the concise and generalizable representation of the "self" identity of a system program by relying on the program's run-time system calls. The learned models were shown to be able to accurately detect anomalies caused by exploits on the system programs. Yet, combination of these two (macro-level and micro-level) for intrusion detection has not been reported.

Intuitively, combination of macro-level and micro-level activities should produce better results as the system can provide multi-layered information. In this paper, we implement a user anomaly detection system based on the intrusion detection framework (Lee, 2001) and conducted several intrusion detection experiments by analysing both macro-level and micro-level activities. User behaviour is modelled by using data mining, and the frequent episode algorithms are used to build the user's normal profiles. At the first step, we gather the user's session data and use frequent episode algorithms to build the user's normal profiles. In the anomaly detection step, we monitor the current user and program activities, and then compare the current activity patterns with the user's profile to determine if the activities are normal or anomaly. The experimental results have shown that the system can detect user anomalies and changes in the user's normal working patterns effectively.

The remaining part of the paper describes the proposed system and is organised as follows. Part 2 explains the process of building user profile. Part 3 discusses the detection model. Part 4 presents the experimental results. Finally conclusion and future work is described.

2 BUILDING USER PROFILE

2.1 Collect the user process data

Our detection system is built on the Windows NT platform using the Win32 library to monitor user and program activities. The *Process Status Helper* (PSAPI.DLL) and the *Tool Help Library* are used to get running process information. Figure 1 shows the system's process monitoring structure (Microsoft, 2002).

When the user starts a new session, the detection system starts automatically and it begins to obtain session information such as login user name, session start time. It then collects information about all running processes in the user's session every five seconds. The major information of running processes collected is presented in table 1. The process identifier is unique and it is assigned by the operating system. Other process information such as memory and input/output (I/O) information reflect the usage of system resources by user processes.

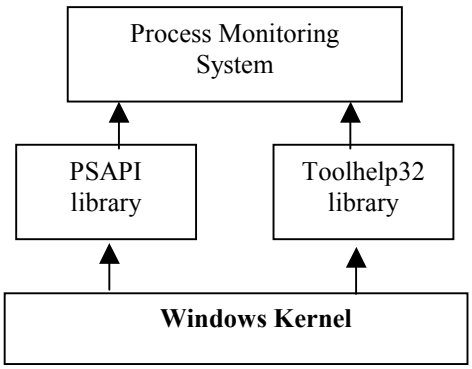

Figure 1: Process monitoring structure

Table 1- Information of monitoring process

Process Attributes	Description
ProcessID	Identifier of process
ProcessName	Name of process
StartTime	Date & Time process started
ExitTime	Date & Time process ended
HandleCount	Number of handles of the process
ThreadCount	Number of running threads of the process
MemoryUsed	Include information about memory used of the process such as current memory used, peak memory used and virtual memory used.
I/O information	Include information about input/output operations such as read, write and other count and transferred data.

2.2 Building user profile

In order to build the user profile, we collect process data from 30 user login sessions. We divide the running process in the system into two types: system processes and user processes. The system processes such as "winlogon.exe" and "services.exe" are processes that are started automatically by the system and they provide basic service to the user processes and user-working environment. User processes such as "acrobat.exe" and "winword.exe" are user application processes that are started by the user in the session. As our purpose is to detect anomalous user activity, we focus on monitoring user processes. We keep track of user activity by recording all related events such as process start, process exit etc. On the program activity level, consumption of system resources by user processes is monitored. Table 2 shows a partial sample of process information collected by the monitoring system.

We applied the *frequent episode algorithms* (Mannila, 1997; Borgelt, 2002) on a collected data set to find the normal usage patterns of a given user. For example, *Alice* is a secretary and she usually uses programs such as email client, web browser and word processor, and her application usage pattern is *Alice(mail:0.95, browser:0.80, word:0.80)*. This means 95% probability of Alice using an email client in her working session and 80% probability of using both web browsers and word processing programs.

Table 2- Process data sample (extract)

SessionID	ProcessID	Process name	Start time
57	884	msimn.exe	2002-10-19 16:17:02
57	748	iexplore.exe	2002-10-19 16:17:12
57	720	winword.exe	2002-10-19 16:19:20
57	156	smss.exe	2002-10-19 16:07:35
57	204	winlogon.exe	2002-10-19 16:07:48
57	232	services.exe	2002-10-19 16:07:50
57	244	lsass.exe	2002-10-19 16:07:51

For each user process, we also need to establish the normal resource usage patterns. One of the most important information we have to evaluate is the number of threads concurrently running, this is called the ThreadCount of a process. An example is shown in Table 1. The more threads a process has the more system resources it uses; for example, if the user runs a web browser, each browser window needs a separate thread. Other information such as handle count, memory usage and I/O information are also processed when constructing the profile.

In general, each user profile contains two parts: (1) the list of user applications with frequency of use and normal start time, and (2) the system resources usage pattern for each user process. In the system resources usage table, each process has an entry and each parameter has its own normal range. For example, Alice's web browser process has a thread count between 6 and 10, memory usage between 12 and 15MB and data transferred (read) between 5 and 10MB. Table 3 shows the system resource table entry for "winword.exe" (a word processor application).

Table 3- Resource usage for winword.exe

Process Attribute	Min	Max
ThreadCount	2	4
WorkingSetSize (bytes)	6582272	12955648
PeakWorkingSetSize (bytes)	6582272	12955648
PagefileUsage (bytes)	2830336	4730880
PeakPagefileUsage (bytes)	2830336	4755456
ReadOperationCount	28	266
WriteOperationCount	2	6646
OtherOperationCount	1363	9330
ReadTransferCount (bytes)	7513	64129
WriteTransferCount (bytes)	162	210906
OtherTransferCount (bytes)	21312	122506

3 ANOMALY DETECTION

The model of our anomaly detection system is shown in figure 2. After the initial step of building the user profile, the detection system is ready for action. When the user starts a new session, the system captures the username and loads the appropriate profile for the user. For testing purposes, we built the user profile as a text file. The monitoring system samples the user processes every five seconds, extracts the information needed and sends them to the detection engine. The detection engine compares process sample information against the user profile and evaluates the result. The comparison procedure is presented in figure 3. Our two-level layered detection scheme assesses the user level activity first, and if the pattern is accepted as normal it goes on to the program level activity pattern test.

In order to increase accuracy and reduce false

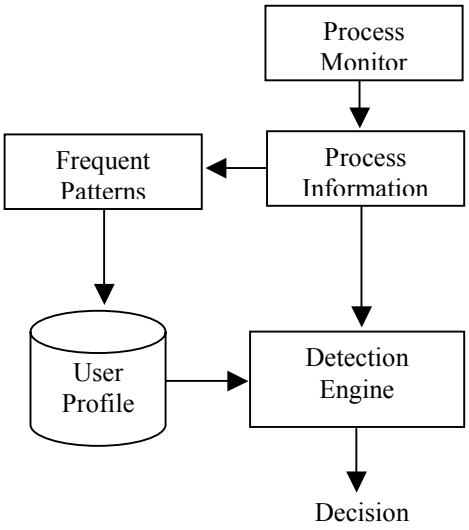

Figure 2: Data mining-based detection model

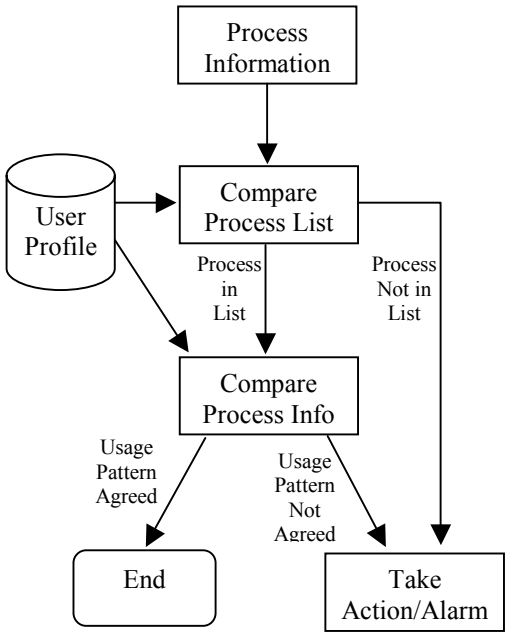

Figure 3: Profile Comparison Procedure

alarm rate, we introduced a window size parameter, which is the minimum period of time in which all comparison results from process samples will be produced. The average of the comparison results is calculated and evaluated, and then the decision about normal/anomalous operation is taken. In our system, the window size is the number of samples taken. It must be chosen carefully, as large window size reduces the false alarm rate but it also increases system response time, and this in turn may make the system miss some anomalous events. In the experiments, we employed a window size of 5 samples. For programming convenience, the data samples were stored in a circular queue.

The actions taken by the detection system can be simply sending an alarm message, or the system may take active steps such as kill the application that causes anomalies or even force the user to log off from the system.

4 EXPERIMENTAL RESULTS

Several experiments were performed on a Windows 2000, 1000MHz CPU machine. The sampling, at a rate of one in every five seconds, had no noticeable effect on the normal tasks in the system. The detection module was in waiting status most of the time (consumed ~0% CPU time) and it only used less than 5% of CPU time when active and processing the samples.

At the user activity level, the system can detect anomalous events such as users running programs that are not in the user normal profile, and a user opening a program at a time that does not fit with the normal time pattern. Table 4 shows samples of Alice's normal and abnormal behaviour that the system can detect. In this experiment, the system captures all session events, such as what programs Alice uses and when they started. The evaluation module first checks whether all of Alice's programs are in the process list. If any of them is not there, the corresponding program is unauthorised, and an anomaly has been detected. If they all are there, the next step is checking the time when it is regularly used. If the timing of this event in Alice's session does not match her profile, it will be considered abnormal.

On the program activity level, we focus on the detection of excessive usage of system resources of user's programs. We monitor eleven parameters of each process, as listed in Table 3, and each one is marked as within or outside the normal range. Human behaviour is difficult to predict, so it is very rare that a user's current activity fully agrees with the stored user profile. To cater for variations, the number of parameters in the normal range is compared to the total number of parameters, a ratio is calculated. We introduced an acceptance threshold whose value is between 0 and 1, and if the ratio is greater than the threshold, the user's current activity is accepted as normal, otherwise it is rejected. Table 5 presents experimental results for this activity level.

Table 4- Alice's normal/abnormal behaviours

Time Start	Application	Normal/ Abnormal
9.00	Mail client	Normal (from 9.00-9.30)
9.20	Web browser	Normal (from 9.00-10.00)
10.00	Word processor	Normal (from 9.30-10.30)
9.10	C++ Compiler	Abnormal (application not in the list)
9.20	FTP program	Abnormal (application not in the list)
16.05	Mail client	Abnormal (not valid time pattern)

Table 5- ThreadCount normal/abnormal ranges

Processes	Normal range	Current Usage	Normal/ Abnormal
msimn.exe (mail client)	6-9	8	Normal
winword.exe (word processor)	2-4	10	Abnormal
iexplore.exe (web browser)	1-17	25	Abnormal

Discussion

On the user level, as Table 4 shows, the system can detect unauthorised use of programs correctly. On the program level, the excessive use of system resources can be detected. In the example shown in Table 5 the ThreadCount parameter indicates the current usage as 8 threads, which is within the normal range of 6-9. However, word processor usage is excessive and so is web browser usage, since ThreadCounts go outside the normal ranges.

We have tested the system with acceptance threshold values of 0.70, 0.75, 0.80 and 0.90 on various user profiles. It was clear that the higher threshold selected the higher the system's accuracy. However, higher threshold may result in false alarms. On the other hand, if the threshold value is too low, the system may miss anomalous events.

5 CONCLUSION AND FUTURE WORK

In this paper an anomaly detection system based on data mining was presented. The frequent episode algorithm was used to get normal usage patterns at the initial step of the system operation. The system is able to detect anomalies or changes in the user's normal working patterns on two levels: on the user level and on the program level. By integrating detection on the user and on the program levels into one system, a good overall picture of a user session can be obtained.

In future, the following improvements are planned for the system.

(i) Dynamic learning of user profiles will allow the system to be more adaptive.

(ii) The range of valid values for a process parameter can change with time, so values for time intervals will be introduced

(iii) The false alarm rate, as well as the undetected anomalies rate depend on the window size. Determining the most suitable window size is an issue for further research.

REFERENCES

Borgelt C., 2002. Finding Association Rules/Hyperedges with the Apriori Algorithm. http://fuzzy.cs.uni-magdeburg.de/~borgelt/software.html#assoc.

Boudaoud, K., Labiod, H., Boutaba, R., and Guessoum, Z., 2002. Network security management with intelligent agents. In the *Network Operations and Management Symposium*, 2000. NOMS 2000. 2000 IEEE/IFIP, 2000. Page(s): 579 -592.

Lane, T., and Brodley, C. E., 1998. Approaches to Online Learning and Concept Drift for User Identification in Computer Security. American Association for Artificial Intelligence press.

Lee, W. and Stolfo, S. J., 2001. A Framework for Constructing Features and Models for Intrusion Detection Systems. In *ACM Transactions on Information and System Security*, Vol. 3, No. 4, November 2000, Pages 227–261.

Lee, W., Stolfo, S. J., and Mok, K. W., Algorithms for System Mining Audit Data. A chapter in *Data Retrieval and Data Mining*, T. Y. Lin and N. Cercone (eds), Kluwer Academic Publishers, 1999.

Mannila, H., Toivonen, H., and Verkamo, I., 1997. Discovery of Frequent Episodes in Event Sequences. In *Data Mining and Knowledge Discovery 1*, pages 259–289. Kluwer Academic Publishers.

Microsoft Corporation, 2002. MSDN Library, http://msdn.microsoft.com/library/default.asp.

Pikoulas, J., Buchanan, W.J., Mannion, M., and Triantafyllopoulos, K. 2001. An agent-based Bayesian forecasting model for enhanced network security. In *ECBS 2001 Proceedings on Engineering of Computer Based Systems – the Eighth Annual IEEE International Conference and Workshop*. Pages 247-254.

WARRENDER, C., FORREST, S., AND PERLMUTTER, B., 1999. Detecting intrusions using system calls: Alternative data models. In the *Proceedings of the 1999 IEEE Computer Society Sympo-sium on Research in Security and Privacy (Berkeley, CA, May)*. IEEE Computer Society Press, Los Alamitos, CA, pages 133–145.

Zhang, R., Qian D., Ba, C., Wu, W., and Guo, X., 2001. Multi-agent based intrusion detection architecture, In the 2001 International Conference Proceedings of *Computer Networks and Mobile Computing*. Page(s): 494 -501.