Process Profiling Using Frequencies of System Calls

Surekha Mariam Varghese,
surekha_laju@hotmail.com
Dept of Computer Sc. and Engg,
M.A. College of Engineering.
Kothamangalam,
India.

K.Poulose Jacob,
kpj@cusat.ac.in
Dept of Computer Science,
Cochin University of Science and Technology,
Kochi,
India.

Abstract

In this paper we discuss our research in developing general and systematic method for anomaly detection. The key ideas are to represent normal program behaviour using system call frequencies and to incorporate probabilistic techniques for classification to detect anomalies and intrusions. Using experiments on the sendmail system call data, we demonstrate that we can construct concise and accurate classifiers to detect anomalies. We provide an overview of the approach that we have implemented.

1. Introduction.

Over the past few years, the scope and importance of security technologies has increased. But many of the modern computer systems are crammed with high security vulnerabilities. Most of the common applications and operating systems are full of security flaws on many levels. These vulnerabilities allow an attacker to gain unauthorized privileges, gain unauthorized access to protected data or interfere with the work of others. Many attacks make use of techniques based on buffer overflows and race conditions.

Detection attempts to compromise the integrity, confidentiality, or availability of computing and communication networks is an extremely challenging problem [1]. Intrusion detection and prevention generally refers to a broad range of strategies for defending against malicious attacks [2]. Intrusion detection can be categorized into misuse detection and anomaly detection.

Misuse detection techniques build signatures of all known intrusions and use these signatures for detecting attempts of intrusions. The main drawback of such systems is that they cannot detect new intrusions whose intrusion patterns are unknown. The need for storing the intrusion signatures for each type of intrusion and the requirement of instant updating of the intrusion signatures impose severe performance bottleneck on misuse detection techniques.

Anomaly based techniques have been useful for Intrusion Detection to detect intrusions without known signatures. However, Anomaly detection techniques suffer from higher false alarm rate compared to misuse intrusion detection techniques. However, although many Anomaly Detection techniques have been proposed to date, no single Anomaly Detection technique can effectively detect all types of intrusions under various scenarios. Anomaly Detection techniques also suffer from high false alarm rate that makes it largely ineffective for Intrusion Detection.

In this paper, the concept of sequence sets is introduced to address the problem. A process can be profiled with frequencies of different system calls in different sequence sets. The representation of a process into different sequence sets and the corresponding frequency distribution has significantly enhanced the detection rate, and lowered the false alarms. To evaluate the effectiveness of the concept, a simple probabilistic model for anomaly detection is proposed.

2. Process Profiling.

Recently, there has been much research on monitoring program behavior to detect intrusions Program-based intrusion detection uses the philosophy that normal program behavior can be characterized in an unambiguous way.

Unlike the behavior of a human user or the behavior of network traffic, the behavior of a program ultimately stems from a series of machine instructions whose meanings we know. The observed programs are usually system programs, and their behavior should not change without our knowledge. Thus, if intrusions can be detected as deviations from normal program behavior, such an intrusion detection technique would be free from false alarms caused by changes in user behavior patterns, and free as well from missed

Second International Conference on Availability, Reliability and Security (ARES'07)
0-7695-2775-2/07 $20.00 © 2007 IEEE

intrusions caused by attackers that mimic benign users [3].

Intrusion detection in such systems is done by comparing the profile of the input process behavior against the normal profile and taking actions according to some predetermined security policies. To profile normal usage patterns, Anomaly detection systems such as IDES[4] makes use of statistical measures on system features like the CPU and I/O activities by a particular user or program. Decision making System features and inter-relationships among different events and features vary highly in different computing environments [5].

System call traces are a common type of audit data collected for performing intrusion detection. A system call trace is the ordered sequence of system calls that a process performs during its execution. The trace for a given process can be collected using system utilities such as strace. System call traces are useful for detecting a user to root exploit or attack. In this type of exploit, a user exploits a bug in a privileged process using a buffer overflow to create a root shell. Typically, the system call trace for a process being exploited is drastically different from the program process under normal conditions. This is because the buffer overflow and the execution of a root shell typically call a very different set of system calls than the normal execution of the program.

Because of these differences, we can detect when a process is being exploited by examining the system calls. Traditionally, these methods typically build models over short contiguous subsequences of the system call trace. There have been many different methods proposed for building models over these short contiguous subsequences.

Reference [6] describes a simple method to determine the normal behavior for privileged processes using local ordering of system calls. Normal sequences are represented with the help of look ahead pairs in [6] and contiguous sequences in [7]. [8] Presents a statistical method for misuse detection by locating sequences, which occur more frequently in intrusion data as opposed to normal data. All these methods predict the probability for a subsequence to belong to a normal process or an exploit. In [4] an alternative representation for system call traces using a bag of system calls is introduced. In [3] and [9] a state based approach is used.

3. Buffer Overflow Attacks.

The most commonly exploited vulnerability in general-purpose operating systems is the buffer overflow. It is encountered due to insufficient bounds checking on arguments that are supplied by users and occurs whenever a request for a buffer access crosses the array / buffer boundary that was allocated for it. For example, after it was first reported many years ago, exploitable "buffer overflow" still exists in some recent system software due to programming errors.

In an overflow attack, the objective of the attacker is to corrupt the information in a carefully designed manner. Commonly they make use of functions that do not check the size of the arguments and pass very large strings as arguments to these functions, which either overwrites the function return address or places an executable code in the stack.

The correct method to prevent such attacks is to provide range checking for arrays or buffers used. Owing to the heavy overhead involved in, it is usually not preferred. For languages like C, where the size of arguments are unknown in most cases, range checking is not a good option. The usual method is to go for static code checking to determine risky functions and to substitute them with safer functions.

Our experiments verify whether buffer overflows alter the execution sequence of a process and attempt to detect anomalies caused by buffer overflows by analyzing system call sequences.

4. Bayesian Networks.

Bayesian Belief Networks have attracted much recent attention as a possible solution for the problems of decision support under uncertainty. They are called Bayesian networks because they make use of bayes rule for probabilistic inference. [10].

The Bayesian network model can represent dependencies among the different objects into its structure. It is made up of a set of variables (nodes) and a set of directed edges between variables. Each node has a number of states and a conditional probabilistic table that describes the probabilistic distribution of the states for the corresponding variable given the states of its parent nodes. Graphically a Bayesian network can be described by a directed acyclic graph [10, 11, 12]. A Bayesian network can effectively represent the dependence between variables and can give a concise specification of the joint probability distribution.

5. Initial Experiments.

Initial experiments were conducted in Redhat Linux on simple processes to verify whether buffer overflows can be detected from system call traces. Buffer overflows were created by passing very large strings containing the intrusion code. System call traces were

collected using the command "strace". A simple C program which is vulnerable to buffer overflow, the buffer overflow code passed to the buffer and the corresponding shell code are given in Figure 1.

Vulnerable C Program

```
int overflow( char *data)
{
char final[100];
strcpy(final, data);
return;
}
main ()
{
char initial[160];
gets(initial);
overflow(initial);
return;
}
```

Overflow Code

```
Jump shellcode
Setreuid(0,0)
Exit(0)
```

Shellcode Used

```
"\x31\xdb\x31\xc9\x31\xc0\xb0\x46\xcd\x80\x31\xdb\x31\
xc0\xb0
\x01\xcd\x80"
```

Figure 1 Buffer Overflow Demonstration

System call traces obtained during normal execution and abnormal execution of the vulnerable program are shown in figure 2. The portion of the system call trace which is altered in the intrusion trace is shown in bold face.

To conduct further experiments, as a commonly used program that is vulnerable to common exploits and buffer overflows, Sendmail daemon was used for studying the normal behaviour and to detect anomalous behaviour. The syslog intrusions [13] are simple examples of buffer overflows in sendmail. Though patches are currently available for the most of the vulnerabilities, sendmail and the buffer overflow attacks on sendmail are good cases for experimental study. Sendmail daemon was examined for detection of buffer overflow attacks.

Normal ExecutionTrace

Execve, Uname, Brk, Old_mmap, Open,Open, Fstat64,Old_mmap, Close, Open,Read, Fstat64 Old_mmap,Old_mmap, Old_mmap, Close, Set_thread_area, Munmap,Mmap2, Read, **Fstat64, Mmap2,Write, Munmap, Exit_group**

Intrusion Trace

Execve, Uname,Brk, Old_mmap, Open,Open, Fstat64,Old_mmap, Close, Open,Read, Fstat64 Old_mmap,Old_mmap, Old_mmap, Close, Set_thread_area, Munmap,Mmap2,Read, **Setreuid, Exit**

Figure 2 Analysis of System Call Trace

In order to construct a good classifier, we need to gather a sufficient amount of training data and identify the set of meaningful features. Due to the unavailability of enough varieties of intrusion trace data, further experiments were conducted using data sets available at University of New Mexico [14].

6. Experiments Using Sendmail

UNM data sets consist of system call traces for many processes. Synthetic data for sendmail, used in the experiments, were collected at UNM on SUN SPARC stations running unpatched SUNOS 4.1.1 and 4.1.4. System calls generated by a process and its children are stored in the same trace. Each trace is a sequence of (process id, system call number). System call numbers are stored in the order in which it is executed. There is a mapping file that associates the system call numbers to the corresponding system call names. The set include normal traces and abnormal traces. A normal trace consists of several invocations of the sendmail program. The abnormal traces used are from syslog-remote intrusion and syslog-local intrusion.

Table 1. A short sequence from sendmail normal dataset

3750 5	3752 105	3752 104	3752 104

The abnormal traces include local and remote intrusions, each with variety commands executed during the attack.

6.1 Data Preparation

System call trace for a particular process is represented as a sequence. Each trace in the data set is a collection of several sequences. Sequences are separated and a frequency chart of system calls for each sequence is prepared. Each sequence is characterized by the start sequence. All sequences with similar starting sequences are grouped into a sequence set. It is assumed that the number of possible normal sequence sets for a particular process is limited. As per the UNM data sets, the number of possible sequence sets for Sendmail is nine and the first seven system calls in the sequence and the frequency of the first system call is used to identify the sequence set to which the particular sequence belongs. Table 2 shows a fragment from frequency chart of sequence set1.

Table 2. A Fragment from the frequency chart of sequence set1 with start sequence 4, 2, 66, 66, 4, 138, 66

System Call Number	Frequency Values				
	PID 3772	PID 3805	PID 3827	PID 3783	PID 3794
1	1	1	1	1	1
2	26	26	26	33	59
3	8	8	8	15	41
4	29	29	29	29	29
5	98	98	98	98	98

6.2 Anomaly Status Determination

Frequency of individual system calls in the execution trace of a process is used for determining the anomaly status of the particular process. Frequency of each system call in the input execution trace is determined and matched with a normal profile. Details of deviations in frequencies of the input execution trace are fed to the Bayesian network. The Bayesian model computes the anomaly score using system call frequencies and prior probability distributions and if the anomaly score is above threshold value, marks it as an anomalous situation.

Anomaly status is determined with the help of a Bayesian network. Frequency chart for the process under consideration is prepared from the input sequence and the corresponding sequence set is determined. A Bayesian Network defines the probability of anomaly for different combinations of system call frequencies. The Bayesian network makes use of a number of model parameters to detect anomalous sequences. The model parameter values are different for different sequence sets.

System call frequencies vary highly in different sequences. Even with in the same sequence set, for certain system calls this variation is unlimited. But for certain system calls, with in the same sequence set, the variation in the frequency is relatively less or limited during normal executions. During a buffer overflow, it is often necessary to insert new code resulting in insertion, deletion or modification of the normal system call sequence. As a consequence, frequency of certain system calls in the sequence deviate from the normal. In most cases, frequencies of system calls can be used to detect anomalous sequence.

System calls are categorized into three groups depending on their frequency variation in anomalous situations. Each model is concerned about a particular category of system calls.

6.3 Model Parameters

Underlying model parameters, their detection mechanisms and significance are described in the following section.

Matching Profile

System calls that has limited or no variation with in the same sequence set are considered in the matching profile model. This model approximates and profiles the distribution of frequencies of system calls during normal executions. The goal of this model is to approximate the distribution of the frequencies of system calls of each sequence set and detect instances that significantly deviate from the observed normal behaviour.

For each sequence set there is a normal profile. The profile stores the minimum for normal and maximum possible deviation for each system call frequency component, for the particular sequence set. Each input sequence is matched with corresponding normal profile. If the frequency components of the input sequence match with the normal profile, with permissible variations, it is treated as a normal sequence by the matching profile model.

Frequency Pattern

Frequencies of certain system calls and subsequences will vary highly even with in the same sequence set. This variation can be considered as normal if this variation is relative to the frequencies of similar system calls. Variation in the frequency of system call Read with system call number 2 in the sequence set1 as shown in table 2 can be considered as example for this case. A Fragment from the frequency

chart of sequence set4, with identifying sequence "105, 104, 104, 106, 105, 108, 112, 1" is listed to demonstrate the variation in frequencies.

Table 3. A Fragment from the frequency chart of sequence set4

System Call Number	1492	1575	1408	1423
2	32	32	12	14
3	15	15	10	11
19	4835	4835	190	670
50	16	16	17	17
78	4814	4814	168	648
104	16052	16052	564	2164
105	9637	9637	344	1304
106	8027	8027	283	1083
108	1610	1610	61	221
112	4830	4830	187	667
128	8	8	10	10

The frequency distribution model captures the concept of a 'normal' system call frequency for such system calls by looking at the relative ranking of the frequency component. It is based on the observation that repeating subsequences will increase the frequency of every system call in the subsequence. The analysis is based only on the relative order of the frequency values and does not rely on the value of the individual system calls. Table 4 lists the ranking of the system call frequencies, as used by the frequency pattern model, corresponding to the processes listed in table 3.

Irregularity count /Presence of system calls

Many of the system calls will not appear in the execution sequence of a particular process and will have zero frequency value. This model takes care of system calls absent in all the normal sequences encountered during training phase. The model examines the input sequence for presence of anomalous system calls and outputs an abnormal value if found.

Once the parameters are correctly identified probability tables can be constructed for predicting the anomaly score. The anomaly score is a value that specifies the extent of the deviation of the received request from the expected profile. It is a compound value that is obtained from the joint probability table. The anomaly score for each request can be in a range from 0.00 to 1.00, where 0.00 represents a completely secure state and 1.00 a sure anomalous state.

Table 4. A Fragment from the frequency pattern chart showing the ranks of system call frequencies of sequence set4

System Call Number	1492	1575	1408	1423
2	8	8	9	9
3	10	10	10	10
19	4	4	4	4
50	9	9	8	8
78	6	6	6	6
104	1	1	1	1
105	2	2	2	2
106	3	3	3	3
108	7	7	7	7
112	5	5	5	5
128	11	11	10	10

6.4 Training

Training involves determination of the sequence sets, system calls used by each of the models, the structure and probabilities associated with each of the nodes in the Bayesian Model. The success of anomaly detection depends on the determination of the correct sequence sets and actual probabilities associated with each of the nodes in the Bayesian Network.

System calls used by each of the models and the sequence sets involved are determined by analyzing the variations in system call frequencies and by matching against the identifying sequence.

The Bayesian Network uses a separate node for each model parameter in each sequence set. The joint probability table associated with a node involving variables X_1 to X_k is estimated from the training data as follows

$$P(X_1 = I_1, \text{---} X_k = I_k) = \frac{N_{X_1=I_1,\text{-----},X_k=I_k}}{N}$$

where $N_{X_1= I_1 ,\text{---} X_k =I_k}$ is the number of observations in which X_1 --. X_k are in states I_1 …. I_k.

7. Performance Evaluation

A concept prototype was developed and implemented to detect buffer overflows. Sequences were identified with their process-ids. Only normal sequences were used for training. Samples were selected using random() function from the list of normal sequences. The prototype was tested using random samples from the list of normal and abnormal sequences. Performance was measured by varying training percentage. For testing, a total of 10,00,000 random data sequences in 10,000 runs were considered in each training category. It was clear from the experiments that frequencies of system calls are good discriminators to detect abnormal behaviour due to buffer overflows. Figure 3 shows the effect of training ratio on false positives and Figure 4 shows the effect of training ratio on accuracy and of true positives.

The system was able to detect all abnormal sequences with a threshold value of 0.25, keeping the number of false positives small. The false positives are caused by system call sequences which significantly deviate from all examples encountered during the training phase. This is due to the huge disparity between the numbers of normal sequences belonging to different sequence sets of the dataset used for evaluation. If this disparity can be removed by selecting all the different varieties of data sequences for training, the number of false positives will be zero.

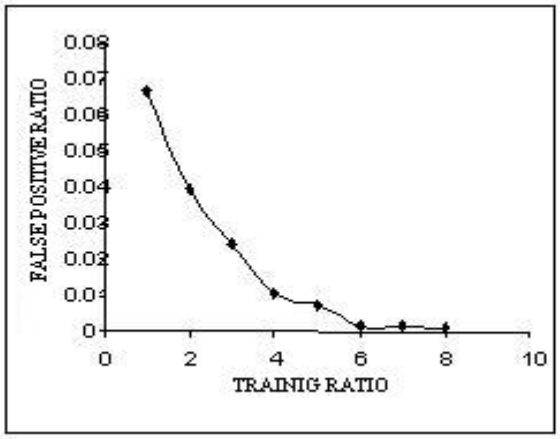

Figure 3 Performance Evaluation 1

The approach can be extended to other processes and for different types of intrusions.

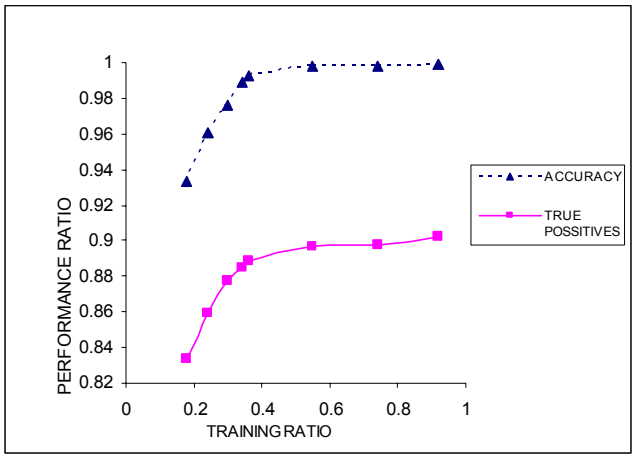

Figure 4 Performance Evaluation2

8. Conclusion

It is a race between intrusion techniques and detection techniques. As more efficient detection techniques are discovered, more complicated intrusion techniques also will evolve. The approach aims at building process profiles with system call frequencies and to detect anomalies by measuring deviations from the process profile. The use of Bayesian network, incorporating different complex possibilities, improves detection and reduces false alarms. The accuracy of the detection models depends on sufficient training data and the right feature set. Preliminary experiments of using the approach on sendmail data provided at the UNM site showed promising results.

9.Acknowledgments

We are very grateful to Anil Somayaji and Stephanie Forrest, for helping us with the necessary intrusion data and providing the details of their experiments at University of New Mexico.

10.References

[1] D.E. Denning, An intrusion-detection model, IEEE Transactions on software Engineering, Vol:13, No:2,pp. 222-232,1987.

[2]. H. S. Javitz and A. Valdes. The SRI IDES Statistical Anomaly Detector. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 1991.

[3] C. C. Michael And Anup Ghosh, Simple, State-Based Approaches to Program-Based Anomaly Detection, ACM Transactions on Information and System Security, Vol. 5, No. 3, August 2002.

[4] Dae-Ki Kang, Doug Fuller, Vasant Honavar, Learning Classifiers for Misuse and Anomaly Detection Using a Bag

of System Calls Representation, In the Proceedings of the 2005 IEEE Workshop on Information Assurance and Security,United States Military Academy, West Point, *NY,,pp 118-125*
http://www.cs.iastate.edu/~honavar/Papers/isi05.pdf
[5].W. Lee, S. Stolfo, and K. Mok. Mining in a Data-flow Environment: Experience in Network Intrusion Detection. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '99)*, San Diego, CA, August 1999.
[6] S. Forrest. A Sense of Self for UNIX Processes. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 120–128, Oakland, CA, May 1996.
[7]. S. A. Hofmeyr, S. Forrest, and A. Somayaji. Intrusion detection using sequences of system calls. Journal of Computer Security, 6:151–180, 1998.
[8] P. Helman and J. Bhangoo. A statistically based system for prioritizing information exploration under uncertainty. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 27(4):449–466, July 1997.
[9] Sekar, R., Bendre, M., Dhurjati, D., And Bollineni, P, A Fast Automaton-Based Method For Detecting Anomalous Program Beha
viors. In *Proceedings Of The 2000 IEEE Symposium On SecurityAnd Privacy*. IEEE Computer Society, Los Alamitos, Calif., 144–155.
[10]. Finn V.Jensen, An Introduction to Bayesian Networks, Springer, 1996.
[11]. J. Pearl. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, 1997.
[12] Probabilistic networks-with-undirected.pdf
[13] CERT Syslog vulnerability-a workaround for sendmail,http://www.cert.org/advisories/CA-95.13.syslog.vul.html,October 19,1995.
[14] Computer Immune Systems - Data Sets and Software http://www.cs.unm.edu/~immsec/systemcalls.htm