# Automatic Generation of Finite State Automata for Detecting Intrusions Using System Call Sequences[*]

Kyubum Wee[1] and Byungeun Moon[2]

[1] Ajou University, Suwon, S. Korea 442-749
kbwee@ajou.ac.kr
[2] SitecSoft, Seoul, S. Korea 135-090
lovtea@sitecsoft.com

**Abstract.** Analysis of system call sequences generated by privileged programs has been proven to be an effective way of detecting intrusions. There are many approaches of analyzing system call sequences including N-grams, rule induction, finite automata, and Hidden Markov Models. Among these techniques use of finite automata has the advantage of analyzing whole sequences without imposing heavy load to the system. There have been various studies on how to construct finite automata modeling normal behavior of privileged programs. However, previous studies had disadvantages of either constructing finite automata manually or requiring system information other than system calls. In this paper we present fully automatized algorithms to construct finite automata recognizing sequences of normal behaviors and rejecting those of abnormal behaviors without requiring system information other than system calls. We implemented our algorithms and experimented with well-known data sets of system call sequences. The results of the experiments show the efficiency and effectiveness of our system.

## 1 Introduction

Intrusion detection techniques can be broadly classified into two classes: misuse detection and anomaly detection. Misuse detection tries to find signatures of intrusion by looking up the known patterns of attack. Anomaly detection maintains normal patterns of behavior and issues an alarm when the system being monitored shows abnormal behavior. Anomaly detection techniques are studied actively, because they have the advantage of being able to detect previously unknown patterns of intrusions.

One of the most important factors of anomaly detection is how to profile normal behaviors. In particular, it should be able to accommodate normal behaviors that are not directly observed while the patterns of normal behavior are collected.

There are many techniques of profiling normal behavior including statistical approach [3, 7, 9, 10, 14], neural networks [2], and Hidden Markov Models (HMM) [17]. Forrest introduced the technique of analyzing system call sequences [4, 6, 17]. It maintains the database of normal sequences of fixed length. It gives an alarm if the difference between the sequence being monitored and the one in the database exceeds the given threshold.

[*] This work was supported partly by the Brain Korea 21 Project and partly by the Institute of Information Technology Assessment research project C1-2002-088-0-3.

There have been studies on modeling normal behavior using finite state automata [8, 12, 16]. The sequences recognized by the finite automaton are considered normal behavior, and those rejected are considered abnormal. The advantages of finite automata over the sequences of fixed length are that they can recognize infinitely many sequences and that they can learn to recognize new patterns through training. However, the procedure of constructing finite automata has been done manually. In this paper, we present algorithms to automatically construct finite automata recognizing normal behavior, and show the performance and efficiency of the system through experiments and analysis.

## 2 Related Works

Forrest et al. introduced the use of system call sequences to model program behaviors [4, 6, 17]. They extract the short sequences of fixed length from the long sequence of system calls generated by processes, called N-grams, and maintain the database of such sequences. When a program is monitored, the N-grams are extracted and matched against the ones in the database of normal sequences. If the miss ratio exceeds the given threshold, a warning is issued. The use of system call sequences has been proven to be a very effective way to intrusion detection. However, since short sequences of fixed length are used, the intruder can dodge detection by carefully inserting spurious system calls within the fixed window size in order not to exceed the threshold.

Wagner et al. use finite automata to represent the result of profiling normal behavior of programs using static analysis of programs [16]. Since this approach analyzes the source code of the program, it has difficulty in modeling various processes generated at runtime.

Sekar et al. examine the program counters where the system calls are made [12]. States and edges of finite automata are labeled by program counters and system calls, respectively. This approach facilitates the construction of finite automata, but has difficulty in stack traversal and dealing with fork/exec to trace program counters.

Kosoresow et al. substitute macros for frequently occurring substrings of the system call sequence, and then construct finite automata recognizing the system call sequences generated by processes [8]. By introducing macros the system call sequences become shorter, and consequently the resulting automaton becomes smaller. This approach has neither the weakness of the N-gram method nor the difficulty of tracing program counters. However, the procedure of selecting macros and constructing finite automata are carried out manually using human insight and intuition. In this paper, we present our study on how to automatically select macros and construct automata.

## 3 Automatic Generation of Finite Automata

We construct finite automata that model normal behavior of programs. In the profiling stage, the sequences of system calls made by the processes are collected. Then a finite automaton recognizing these sequences is constructed. Once the finite automaton is constructed, it is used to monitor the executions of the program. If the sequences of the system calls made by the processes are accepted by the finite automaton, then the