침입탐지를 위한 유한상태기계의 생성 기법

임 영 환[†] · 위 규 범[††]

요 약

침입 탐지 기법에 있어서 유한상태기계(finite automata)를 통해 정상 행위를 프로파일링 하는 연구들이 많이 진행되어 왔으나, 자동으로 간결한 형태의 오토마타를 생성하는 것이 매우 어려웠다. 이전 연구에서는 프로세스를 오토마타로 프로파일링 하기 위해 빈번한 시스템 콜 서열(system call sequence)을 매크로(macro)로 치환하고, 이러한 서열을 인식하는 오토마타를 수작업으로 생성하였다. 본 논문에서는 이러한 오토마타를 자동적으로 생성할 수 있도록, 서열 정합(sequence alignment)을 수행하고 스트링으로부터 반복되는 패턴들을 찾아내어 프로세스들로부터 매크로를 추출하고 오토마타를 생성해내는 방법을 제안한다. 생성된 오토마타가 침입탐지에 효과적으로 이용될 수 있음을 실험을 통하여 보였다.

Generation of Finite Automata for Intrusion Detection

Younghwan Lim[†] · Kyubum Wee[††]

ABSTRACT

Although there have been many studies on using finite automata for intrusion detection, it has been a difficult problem to generate compact finite automata automatically. In a previous research an approach to profile normal behaviors using finite automata was proposed. They divided the system call sequence of each process into three parts : prefix, main portion, and suffix, and then substituted macros for frequently occurring substrings. However, the procedure was not automatic. In this paper we present algorithms to automatically generate intrusion detection automata from the sequence of system calls resulting from the normal runs of the programs. We also show the effectiveness of the proposed method through experiments.

키워드 : 침입 탐지(Intrusion Detection), 비정상행위 탐지(Anomaly Detection), 프로세스 행위 프로파일링(Process Behavior Profiling), 유한상태기계(Finite Automata)

1. 서 론

최근 활발히 연구되고 있는 침입탐지시스템은 정상 행위를 프로파일링(profiling)하고 이 프로파일링된 행위로부터 벗어나는 것을 침입으로 간주하는 비정상 행위 탐지기법(anomaly detection)과, 침입 패턴을 놓고 이 패턴과 일치하는 것을 침입으로 판정하는 오용 탐지기법(misuse detection)으로 나뉜다. 이 두 가지 기법 중에서 비정상 행위 탐지 기법은 새로운 형태의 침입을 찾아낼 수 있는 장점을 가지고 있다는 점에서 많은 연구자들의 관심을 끌고 있다. 이 비정상탐지기법의 가장 중요한 부분은 어떻게 정상 행위를 정의하고 프로파일링 하는가 이다[1]. 특히 가능한 모든 정상 행위를 프로파일링 하는 것이 불가능하기 때문에 정상 행위를 프로파일링 한 결과는 프로파일링 과정에서 사용되지 않은 다양한 정상 행위들을 수용할 수 있어야 한다.

정상 행위를 프로파일링 하기 위해서 통계적 방법을 이용하거나 인공신경망(neural network) 또는 Hidden Markov Model(HMM) 등을 사용하기도 한다[1]. 특히, Forrest에 의해 제안된 방식은 시스템 콜(system call)을 고정된 길이의 여러 문자열로 분리하여 고안된 알고리즘을 통해 데이터베이스에 입력해 놓고, 이후 감시하는 프로세스의 시스템 콜 서열(sequence)이 데이터베이스 안에 존재하지 않는 경우의 횟수를 세어 횟수가 정해진 임계값(threshold)을 넘을 경우 이를 침입으로 판정한다. 이 방식은 프로그램의 중요한 실행 단위인 시스템 콜을 이용한 시도라는 점에서 의의가 있다[2, 4].

Finite automata(FA)를 이용하여 정상 행위를 모델링 하는 기법은 FA가 한정된 저장공간에 비해 임의의 길이의 무한히 많은 서열을 인식할 수도 있으며, 루프나 분기 등에 의해 학습에 이용된 패턴 이외에 새로운 형태의 행위가 나타날 수 있는 등의 장점이 있기 때문에 많이 연구되어 왔다[3, 5-7]. 그러나, 간결한 형태의 FA를 학습하는 것이 매우 어려워, 대부분의 이전 연구에서 제시한 오토마타는 사

※ 이 논문은 2002년도 두뇌한국21사업과 정보통신부의 정보통신학술기초연구사업으로 지원되었음.
† 정 회 원 : 라딕스 연구소 연구원
†† 종신회원 : 아주대학교 정보 및 컴퓨터공학부 교수
논문접수 : 2002년 8월 26일, 심사완료 : 2003년 3월 10일

용자의 개입이 상당 부분 작용했었다. 이후, FA와 유사한 HMM을 이용한 연구가 진행되었으나 학습에 지나치게 큰 오버헤드가 있으며, 오버헤드에 비해 비교적 적은 탐지율의 향상을 가져왔다[11].

본 논문에서는 침입 탐지에 이용할 수 있도록 시스템 콜 서열의 정상 행위를 모델링하고, 기존의 FA를 통한 모델링 기법들의 단점을 극복할 수 있는 자동화된 FA 생성 알고리즘을 제시한다.

2. 연구의 배경

Forrest의 연구에서는 실행되는 프로세스에 의해 발생되는 시스템 콜 서열을 특정 길이로 중첩되도록 짧게 자른 시스템 콜 서열들의 집합을 정상 행위로 정의하였다. 시스템 콜 서열들로부터 순서적인 정보를 데이터베이스에 입력하고 이후 프로세스를 모니터링 할 때, 프로세스의 시스템 콜의 짧은 부분서열들이 데이터베이스에 존재하지 않는 경우의 비율을 계산하여 침입인지 아닌지를 판별하였다[4].

FA를 이용하여 정상행위를 프로파일링하는 다른 연구들로는 우선 시스템 콜이 아닌 정적분석(static analysis)을 이용한 것이 있다[6]. 이 방법은 프로그램의 소스 코드로부터 함수의 콜 그래프 모델인 transition diagram을 생성하였다. 그러나, 프로그램의 소스를 기반으로 설계되므로 프로그램의 실행시에 발생하는 다양한 프로세스들을 포함하는 정상 행위를 모델링하기에 어렵다.

Sekar 등은 시스템 콜과함께 시스템 콜이 호출된 스택 포인터(stack pointer)를 참조하여 같은 프로그램 카운터(program counter)를 가지는 시스템 콜인 경우에 같은 상태(state)로 간주하는 방식으로 FA를 생성한다[3]. 이러한 방식은 시스템 콜을 추적(trace)할 때, 스택을 조사해야 하는 어려움이 따른다.

Michael 등의 연구에서는 트레이닝 데이터가 많다 하더라도 정상 프로그램의 행위로부터 간단한 방식으로 자동적으로 생성된 FA와 통계적인 수치 두 가지를 이용하여 침입을 탐지하는 방식을 제안하였다[7].

기존의 시스템 콜을 이용한 침입 탐지 기법들은 trace들의 길이가 각각 다르고, 실제 비정상적인 시스템 콜이 발생하는 것이 특정 구역에 뭉쳐져 있기 때문에, 대부분 전체 trace를 locality frame으로 나누고, locality frame 내에 threshold를 넘는지를 확인함으로써 침입을 판별한다. 그러나, 이러한 경우 침입자가 각 locality frame에서 threshold를 넘지 않도록 침입행위 안에 충분히 많은 부가적인 시스템 콜을 삽입함으로써 탐지를 피할 수 있다.

이와 같은 취약성을 없애기 위해 Kosoresow는 locality frame을 사용하지 않고, 각 프로세스의 전체 행위를 프로파일링 하는 방법을 제안하였다. 그는 다음의 단계들을 거쳐서 프로세스의 행위를 하나의 finite automaton으로 표현하였다[5].

제 1단계 : 시스템 콜 trace를 프로세스 별로 분리하고, 시스템 콜 서열이 유사한 프로세스들을 묶어서 몇 개의 카테고리로 분류한다.

제 2단계 : 각 카테고리별로 공통된 prefix와 suffix를 찾아서 프로세스들을 prefix, main part, suffix의 세 부분으로 분할한다.

제 3단계 : 서열에 자주 나타나는 패턴을 매크로로 정의하고, 이러한 패턴들을 매크로로 대체한다.

제 4단계 : 모든 프로세스의 시스템 콜 서열을 인식하는 오토마타를 구성한다.

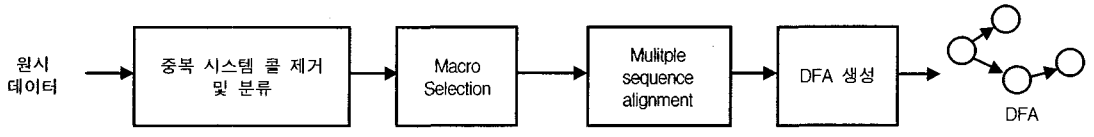
이렇게 만들어진 침입탐지 모델에서는 대부분의 매크로가 긴 서열이므로 정상적인 짧은 서열을 조합하여 침입하는 경우를 탐지할 수 있다. 그러나, 여기서 매크로를 정의하는 부분이나 FA를 생성하는 과정이 자동화되어 있지 않았으므로 대량의 데이터를 분석하여 정상 행위의 패턴을 프로파일링 하기 어렵다.

3. 자동화된 DFA 생성 기법

본 연구에서는 프로세스의 정상행위를 프로파일링 하기 위하여 시스템 콜을 사용하였다. 특히, [5]에서 제안한 프로세스들의 서열을 하나의 오토마타로 표현하는 방법을 이용하며, 여기에 대량의 데이터를 위한 자동화 기법을 추가하였다. 본 논문에서 제안하는 FA의 생성은 (그림 1)과 같은 단계를 거치게 된다. 이렇게 생성된 FA는 정상행위를 정규언어(regular language)로 표현한 것이며, 프로세스의 시스템 콜 서열이 이 FA에 의하여 인식(accept)되는 경우는 정상행위로, 인식되지 않는 경우(reject)는 비정상행위로 판정한다.

3.1 중복 서열 제거 및 분류

FA를 생성하기 위해 최초로 주어지는 데이터는 프로세스 별로 분리된 시스템 콜의 서열이다. 실제로 프로파일링



(그림 1) DFA 생성과정

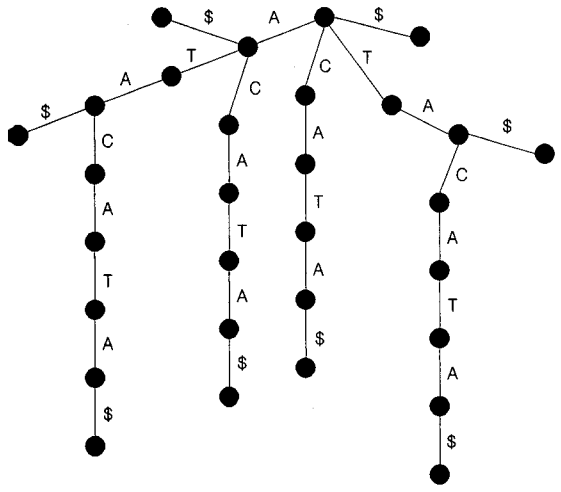
해야 하는 것은 프로그램의 행위지만, 실제로 프로세스 사이의 관계를 모두 파악하는 것이 매우 어려우므로 각각의 프로세스를 분리해서 프로세스 단위로 프로파일링을 진행한다[5].

분리된 프로세스들 중에는 다른 프로세스와 동일한 서열을 가지는 프로세스가 존재하기 때문에 우선 동일한 서열을 가지는 프로세스들 중에서 하나의 프로세스만 남도록 제거한다.

프로세스를 제거하기 위해서 trie를 사용한다. Trie는 하나의 노드에 하나의 알파벳이 저장되는 자료 구조로서 prefix가 동일한 경우 같은 노드로 합쳐지게 된다[8]. 입력으로 주어진 프로세스들로부터 trie를 만들면, 자연히 동일한 시스템 콜 서열을 가지는 프로세스는 제거된다.

Trie를 통해 동일한 프로세스를 제거한 후에 유사한 프로세스들의 그룹으로 분류한다. Trie를 생성하면 프로세스의 prefix에 따라 자동적으로 그룹이 묶여지게 되며, prefix에 나타나는 시스템 콜의 종류와 빈도를 통해 분류한다.

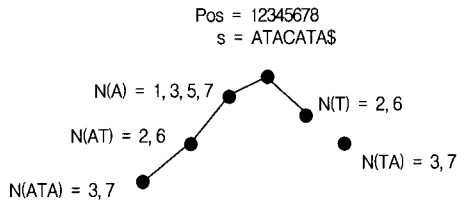
3.2 매크로의 선택 및 설정



(그림 2) 스트링 ATACATA의 suffix trie

매크로를 추출하기 위해서 [9]에서 제안된 suffix trie를 사용하였다. Suffix trie는 주어진 문자열의 맨 끝에 특수문자 $를 덧붙인 후에 구성한다. Suffix trie에서 루트노드(root node)로부터 각 단말노드(leaf node)까지의 경로는 주어진 문자열의 각 suffix를 나타낸다. 따라서 루트노드로부터 각 중간노드까지의 경로는 각 부문자열(substring)을 나타낸다. 각 노드는 그 노드가 나타내는 부문자열의 끝나는 위치를 저장하고 있다. (그림 2)는 suffix trie의 예이다. [9]의 알고리즘을 이용하여 주어진 횟수 이상으로만 나타나는 부문자열들을 추출하는 것이 가능하다. (그림 3)은 (그림 2)에서 발생 빈도가 2회 이상인 부문자열들만의 suffix trie를 나타낸다.

매크로를 자동으로 추출하기 위해서는 우선 어떤 매크로를 선택할 것인지에 대해서 기준이 필요하다. 가장 이상적인 것은 가능한 한 적은 수의 매크로를 사용하여 문자열을 되도록 짧게 만들 수 있는 매크로의 집합을 찾는 것이다. 그러나, 최적의 매크로 집합을 찾는 것은 계산 복잡도가 매우 높으므로, 본 연구에서는 매크로를 하나씩 개별적으로 평가하면서 매크로를 선택하였다. 하나의 매크로는 그 매크로가 나타내는 부문자열을 매크로로 치환하였을 때, 전체 문자열의 길이가 얼마나 짧아지는가에 의하여 평가한다. 하나의 매크로가 나타내는 부문자열의 길이를 L이라 하고 그 부문자열이 전체 문자열에서 나타나는 회수를 K라 하면, 그 문자열을 매크로로 치환하였을 때 전체 문자열의 길이는 R = K * (L - 1) 만큼 감소한다.



(그림 3) 빈도를 2회 이상으로 제한했을 때의 스트링 S의 suffix trie. N(s)는 S의 substring의 위치의 집합

매크로를 선택하는 절차는, 프로세스의 시스템 콜 서열들을 모두 연결한 긴서열 S의 첫 문자부터 시작하며, 그 다음 문자들을 하나씩 추가하면서 테스트를 수행한다. 주어진 부문자열이 나타나는 위치들을 suffix trie로부터 찾을 수 있으므로, 부문자열의 위치들을 찾아 매크로에 의해 치환되는 영역을 구한다. 이에 따라 각 부문자열을 위에서 설명한 방식대로 평가한다. 평가값이 증가하는 동안에는 계속해서 부문자열의 길이를 증가시켜 나아가다 평가값이 감소하는 지점에서 멈추어서, 그 이전까지의 부문자열을 하나의 매크로로 지정한다. 정지한 지점에서부터 다시 위의 과정을 반복하여 다음 매크로를 지정한다. S의 마지막 문자를 검사할 때까지 이러한 과정을 반복한다. (그림 4)는 문자열이 ATACATA인 경우에, 위 알고리즘을 적용한 예이다. (그림 3)에서처럼 제 3단계에서 R이 4이고 다음 단계에서는 3이므로, 제 3단계의 ATA를 매크로로 선택하고, 스트링의 네 번째 위치에서 다시 같은 과정을 반복한다. 이때, 이전의 매크로에의해 이미 선택된 위치는 새로운 매크로에 의해 다시 선택되지 않도록 주의한다.

1. S = A T A C A T A $ R = 4 * (1 - 1) = 0
2. S = A T A C A T A $ R = 2 * (2 - 1) = 0
3. S = A T A C A T A $ R = 2 * (3 - 1) = 0
4. S = A T A C A T A $ R = 1 * (4 - 1) = 0
5. S = A T A C A T A $ R = 1 * (1 - 1) = 0

(그림 4) 매크로 설정의 예

3.3 다중 서열 정합(Multiple sequence alignment)

택한 다음에는, 같은 매크로가 주어진 횟수 이상 반복되는 경우 이것을 Kleene closure 표현으로 나타낸다. Kleene closure (또는 star closure)이란, x* = {Λ, x, xx, xxx, xxxx, ····· }와 같이 특정 문자가 0번 이상 반복되는 모든 문자열의 집합을 나타낸다. 본 연구에서는 4회 이상 반복되는 문자를 그 문자의 Kleene closure 표현으로 대체하였다.

반복되는 문자를 Kleene closure로 치환한 다음에는, 프로세스들의 시스템 콜 서열의 집합에 multiple sequence alignment를 수행한다. multiple sequence alignment는 (그림 5)와 같이 서로 다른 문자열들이 같은 문자열에서 파생되었다고 가정하고 이들을 정합하는(align) 작업이다. 하나의 문자열이 다른 문자열로부터 파생되는 과정은 문자 하나의 삽입, 문자 하나의 삭제, 또는 문자 하나를 다른 문자로 대체하는 세 가지 연산의 반복적인 적용을 통하여 이루어진다고 가정한다. Multiple sequence alignment는 주어진 문자열들을 정합하여 이들 삽입, 삭제, 대체 연산의 횟수를 최소화하는 작업이다.

```
S1 = ATTGCCATT        ATTGCCATT--
S2 = ATGGCCATT        ATGGCCATT--
S3 = ATCCAATTTT       ATC-CAATTTT--
S4 = ATCTTCTT         ATCTTC-TT--
S5 = ACTGACC          ACTGACC--
```

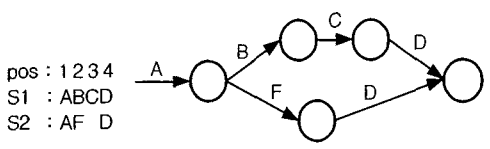
(그림 5) Multiple sequence alignment의 예

Multiple sequence alignment를 수행하는 것은, 프로세스들의 시스템 콜 서열들을 모두 인식하는 FA를 구성하기 위한 전처리 작업으로서의 의미를 가진다. 최적의 sequence alignment를 구하는 일은 계산복잡도가 매우 높으므로, 본 연구에서는 star alignment heuristic을 사용하였다[10].

3.4 오토마타의 생성

본 연구에서는 다중 서열 정합된 결과로부터 다음과 같은 규칙에 의하여 FA를 생성하였다 : *Multiple sequence alignment의 결과에서 같은 위치에 있는 같은 문자는 같은 상태로 전이한다.*

예를 들어서 (그림 6)에서와 같이 ABCD와 AFD가 정합되어 있을 때, 두 문자열의 D는 모두 4번째 자리에 위치해 있으므로 문자 D는 같은 상태로 전이하는 두 에지(edge)의 레이블(label)이 된다.



(그림 6) DFA 생성의 예

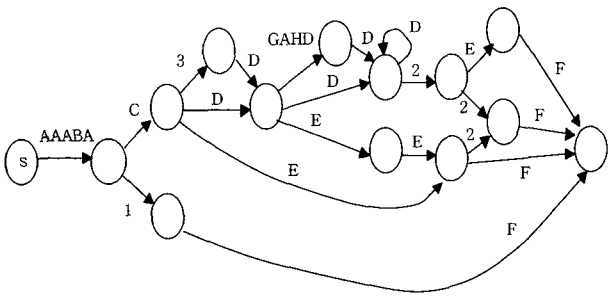
4. 실 험

실험 데이터는 Forrest 등의 연구[2]에서 사용한 데이터 중에서 프로세스의 수가 비교적 많은 sendmail과 lpr의 시스템 콜 서열 데이터를 이용하였다. 원시 데이터는 프로세스 ID와 시스템 콜의 고유번호의 쌍이 연속된 긴 서열로 이루어져 있다. 먼저 원시 데이터로부터 각 프로세스 ID에 해당하는 시스템 콜 서열을 추출한다. 이들 중에는 완전히 동일한 시스템 콜 서열을 가지는 프로세스 ID들이 존재한다. 이러한 경우에는 하나의 프로세스 ID의 시스템 콜 서열만 남겨 놓고, 나머지 동일한 서열들은 제거한다. 동일한 서열을 찾아내는 작업은 trie에서 탐색을 통하여 쉽게 수행할 수 있다.

위와 같이 프로세스들의 시스템 콜 서열들을 trie에서 탐색하여 보면, 서열들은 prefix에 따라 세 개의 그룹으로 나뉘어진다. 세 그룹에 속하는 서열들은 각각 길이가 7, 32, 125인 동일한 prefix를 가진다. 각 그룹에 속하는 서열들을 하나의 긴 문자열로 연결하고 매크로 선택 알고리즘을 수행한다. 이때, 프로세스가 연결되는 부분마다 문자열의 끝을 나타내는 특수문자($)를 추가하여, 하나의 프로세스의 뒷부분과 뒤에 이어지는 다음 프로세스의 앞부분이 연결된 매크로가 선택되지 않도록 한다. 이어서 앞의 제 3절에서 설명한 과정을 수행하여, 각 그룹에 속하는 서열들을 인식하는 오토마튼을 생성한다. <표 1>은 (그림 7)의 각 매크로의 정의이며, (그림 7)은 첫 번째 그룹에 속하는 서열들을 인식하는 FA이다.

〈표 1〉 매크로 정의

매 크 로	시스템 콜 시퀀스
A	105 104 104 106
B	5 4 5 5 40 40 41
C	61 5 85 50 27 18 50 17 2 3
D	2 3 3
E	3 2
F	3 5 6 6 112 112 19 128 9 9 5 9 9 5 112 4 5 5 5 5
G	5 6 112 112 19 128 41
H	61 5 50 27 18
I	61 5 85 50 27 18 50 27 2 2



(그림 7) sendmail의 첫 번째 그룹에서 생성된 DFA

이러한 과정을 거쳐서 생성된 FA가 감시하는 프로세스의 시스템 콜 서열을 인식하는 경우에 그 프로세스가 정상적인 행위를 보인다고 판정하며, 인식하지 않을 경우에 해당 프로세스를 침입으로 판정한다.

<표 2>는 sendmail과 lpr을 가지고 FA를 생성했을 때의 정상행위와 비정상행위 모두에 대한 테스트 결과이다.

테스트에는 Forrest의 데이터를 사용하였다[12]. <표 2>에서 sendmail(CERT)의 정상행위 데이터는 147개의 시스템 콜 서열로 이루어져 있다. 이것은 많은 양의 데이터가 아니므로 실험의 정확도를 높이기 위하여 다음과 같은 테크닉을 사용하였다. 이들을 삼등분하여 각 49개의 서열을 가지는 세 그룹으로 나누었다. 세 그룹 중에서 두 개의 그룹을 가지고 FA를 생성하는 training 과정에 사용하였고, 나머지 한 개의 그룹을 가지고 test에 사용하였다. 이러한 작업을 조합을 달리하여 세 번 실험을 한 결과가 <표 2>의 sendmail(CERT)의 training과 정상행위의 test에 대한 결과이다. 이 경우에 비정상행위에 관한 test는 주어진 18개의 비정상행위 데이터를 그대로 사용하였다.

lpr(MIT)의 경우는 2766개의 정상행위 서열 중에서 1000개를 무작위로 선택하여 training에 사용하였으며, 나머지 중에서 500개를 무작위로 선택하여 test에 사용하였다. 이러한 작업을 두번 반복하여 training과 정상행위의 test에 대한 결과를 얻었다. 비정상행위에 대한 테스트는 주어진 1001 비정상행위 서열을 사용하였다.

lpr(UNM)의 경우는 정상행위 데이터가 9개로서 너무 적으므로, 이들 데이터를 모두 training에 사용하였으며 테스트에는 사용하지 않았다. 비정상행위에 대한 테스트는 주어진 1001 비정상행위 서열을 사용하였다.

lpr 프로그램이 sendmail보다 규칙적인 동작을 보이기 때문에 정상행위를 인식하지 못하는 경우가 sendmail에 비해 적었다. 비정상행위의 경우는 sendmail과 lpr 둘다 모든 프로세스가 FA에서 거부되어 100%의 탐지율을 보였다. 정상행위의 경우에는 sendmail은 5%, lpr은 0.7%의 긍정적 결함(false positive)을 보였다.

5. 결 론

오토마타가 정상행위를 프로파일링 하는데 있어서 많은 장점을 가지고 있음에도 불구하고, 잘 사용되지 못한 이유는 정상 데이터로부터 자동으로 오토마타를 생성할 수 있는 효율적인 방법이 없기 때문이었다.

본 연구에서는 suffix trie를 통해서 프로세스들의 시스템 콜 서열로부터 매크로를 추출하고, 다중서열정합(multiple sequence alignment)를 이용하여 자동으로 DFA를 생성하는 방법을 설계 구현하고 실험하였다.

본 연구에서 FA를 생성하는 거의 대부분의 과정을 자동화 할 수 있었고, sendmail 프로그램과 lpr 프로그램에 대하여 실험해 본 결과, 비정상행위인 경우에는 FA에 의하여 거의 모두 거부(reject)되었으며, 정상행위의 경우에 약간의 긍정적 결함이 존재하기는 하지만 대부분이 FA에 의하여 인식(accept)됨으로써 침입탐지 알고리즘으로서의 가능성을 충분히 보여주었다.

본 논문에서 제시한 방법은 [5]와 마찬가지로 프로세스의 시스템 콜 서열 이외의 다른 어떤 정보도 필요로 하지 않기 때문에, 모니터링시에 시스템에 부담을 주지 않으며, FA를 다항식(polynomial) 시간 안에 자동으로 생성할 수 있고, 탐지하는 속도 면에서도 단지 시스템 콜 하나에 대하여 FA에서 한번의 전이만 일으키면 되기 때문에 선형(linear) 시간에 탐지가 가능하다.

서열의 개수를 k, 가장 긴 서열의 길이를 m, 모든 서열의 길이의 합을 n이라 할 때, suffix trie를 구성하는 알고리즘은 $O(n^2)$의 시간복잡도를 가진다[9]. 동일한 서열들을 제거하고 서열들을 그룹핑하는 작업은 suffix trie에서 $O(n)$의 시간이 걸림을 쉽게 알 수 있다. 매크로를 설정하는 작업은, 최악의 경우에 서열의 각 문자에 대해서 전체 서열을 검사해야 하므로, $O(n^2)$이다. 또한 다중서열정합의 star alignment 휴리스틱 알고리즘의 시간복잡도는 $O(km^2+k^2l)$이다 [10]. 여기서 l은 align-ment가 이루어진 결과의 길이이다. 최악의 경우에 l은 모든 서열의 길이의 합이 되므로 n이 l의 상한이다. 따라서 오토마타의 생성 과정은 $O(n^2+km^2+k^2n)$의 다항식 시간에 이루어짐을 알 수 있다.

본 연구에서는 매크로 선정과 서열의 정합 과정은 완전히 자동화되어 있으나, 오토마타 생성 부분은 완전히 자동화되어 있지 않다. 오토마타 생성 과정도 완전히 자동화하는 것이 향후 연구의 과제이다. 또한 현재의 매크로 선정 절차는 서열의 앞 부분부터 찾아 나아간다. 그러나 이 방법은 일단 매크로의 일부로 설정된 부분은 다른 매크로의 일부로 재고되지 못함으로서, 서열의 길이를 최소화하지 못한

〈표 2〉 sendmail과 lpr의 정상행위와 비정상행위에 대한 실험결과

Trace 데이터 (수집기관)	Training에 사용된 프로세스의 수	Test	
		Normal	Abnormal
		FA에 의하여 거부된 프로세스의 수 / Test에 사용한 정상행위 프로세스의 수	FA에 의하여 거부된 프로세스의 수 / Test에 사용한 비정상행위 프로세스의 수
sendmail(CERT)	294	8/147	18/18
lpr(MIT)	2000	7/1000	1001/1001
lpr(UNM)	9		1001/1001

다. 매크로 선정 절차의 효율성을 높이는 것도 향후 연구 과제이다.

본 연구에서 제시한 방법은 빠른 시간에 오토마타를 생성할 수 있고 생성된 오토마타의 크기도 작은 매우 효율적인 방법이며, 정상행위의 프로파일링을 위하여 오토마타를 자동으로 생성하는 방법에 대한 최초의 연구이다.

참 고 문 헌

[1] R. Bace, "Intrusion Detection," Macmillan Technical Publishing, pp.91-117, 2000.

[2] S. Hofmeyr and S. Forrest, "Intrusion Detection using Sequences of System Calls," Journal of Computer Security Vol.6, pp.151-180, 1998.

[3] R. Sekar and M. Bendre, "A Fast Automaton-Based Methodfor Detecting Anomalous Program Behaviors," Proceeding of the 2001 IEEE Symposium on Security and Privacy, pp.144-155, 2001.

[4] S. Forrest, "A Sense of Self for Unix Process," Proceedings of the 1996 IEEE Symposium on Security and Privacy, IEEE Computer Society Press, pp.120-128, 1996.

[5] A. Kosoresow, "Intrusion Detection via System Call Traces, IEEE Software," Vol.14, No.5, pp.35-42, 1997.

[6] D. Wagner, "Intrusion Detection via Static Analysis," Proceedings of the 2001 IEEE Symposium on Security and Privacy, pp.156-169, 2001.

[7] C. Michael, "Two State-Based Approaches to Program-based Anomaly Detection," Proceedings of 16th Annual Computer Security Applications Conference, Conference, pp.21-30, 2000.

[8] A. Aho, "Data Structures and Algorithms," Addison Wesley Publishing, pp.163-169, 1983.

[9] J. Vilo, "Discovering Frequent Patterns from Strings," Department of Computer Science, University of Helsinki, Technical Report C-1998-9, May, 1998.

[10] S. Carlos, "Introduction to Computational Molecular Biology," PWS Publishing Company, pp.49-80, 1997.

[11] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting Intrusions using System Calls : Alternative Data Models," Proceedings of the IEEE Symposium on Security and Privacy, pp.133-145, 1999.

[12] http://www.cs.unm.edu/~immsec/systemcalls.htm.

임 영 환

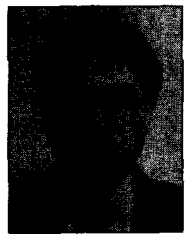
e-mail : yhlim@radix.co.kr

2000년 아주대학교 정보및컴퓨터공학부 (학사)

2002년 아주대학교 정보통신전문대학원 정보통신공학과(석사)

2002년~현재 라딕스 연구소 연구원

관심분야 : 계산이론, 임베디드시스템, 소프트웨어공학



위 규 범

e-mail : kbwee@madang.ajou.ac.k

1978년 서울대학교 수학과(학사)

1984년 University of Wisconsin 전산학과(석사)

1992년 Indiana University 전산학과 (박사)

1993년~현재 아주대학교 정보 및 컴퓨터공학부 교수

관심분야 : 컴퓨터 이론