2001년 한국정보처리학회 추계 학술발표논문집 제8권 제2호

Deterministic Finite Automata를 이용한 프로세스 행위 모델링

임영환, 위규범
아주대학교
e-mail : yhlim@netsgo.com , kbwee@ajou.ac.kr

Process Behavior Modeling Using Deterministic Finite Automata

Young-Hwan Lim, Kyu-Bum Wee
Dept. of Information and Communication, Ajou University

요 약

침입 탐지 기법에 있어서 finite automata 를 통해 정상 행위를 프로파일링 하는 연구들이 많이 진행되어 왔으나, 자동으로 간결한 형태의 오토마타를 생성하는 것이 매우 어려웠다. 이 논문에서는 많은 침입 탐지 기법의 데이터 소스로 사용되고 있는 시스템 콜을 이용하여 자동으로 finite automata 를 생성하고, 여기에 언어 압축 알고리즘을 이용하여 오토마타를 압축하고 일반화 시킴으로써 다양한 프로세스의 행위들을 프로파일링 하도록 하였다. 제안된 알고리즘을 통해 모델링 한 후 정상 행위와 비정상 행위를 가지고 실험을 한 결과 이들 사이에는 많은 수치적인 차이가 있음을 발견하였고, 이 결과를 바탕으로 침입을 탐지하는 것도 충분히 가능함을 알 수 있었다.

1. 서론

최근 활발히 연구되고 있는 침입 탐지 시스템은 정상 행위를 프로파일링 하고 이 프로파일링된 행위로부터 벗어나는 것을 침입으로 간주하는 비정상탐지기법과, 침입 패턴을 놓고 이 패턴과 일치하는 것을 침입으로 판정하는 오용탐지기법으로 나뉜다. 이 두 가지 기법 중에서 비정상 행위탐지 기법은 부정적 결함이 적어 새로운 형태의 침입을 찾아낼 수 있는 장점을 가지고 있다는 점에서 많은 연구가들의 관심을 끌고 있다. 이 비정상탐지기법의 가장 중요한 부분은 어떻게 정상 행위를 정의하고 프로파일링 하는가이다[1].

정상 행위를 프로파일링 하기 위해서 통계적 방법을 이용하거나 neural network, Hidden Markov Model(HMM) 등을 사용하기도 한다[1]. 특히, Forrest에 의해 제안된 방식은 시스템 콜을 고정된 길이의 여러 문자열로 분리하여 고안된 알고리즘을 통해 데이터 베이스에 입력해 놓고, 이후 감시하는 프로세스의 시스템 콜 시퀀스가 데이터 베이스 안에 존재하지 않는 경우의 횟수를 세어 침입을 탐지한다. 이 방식은 프로그램의 중요한 실행 단위인 시스템 콜을 이용한 시도라는 점에서 의의가 있다[2].

Finite automata를 이용하여 정상 행위를 모델링 하는 기법은 finite automata가 한정된 저장공간에 비해 임의의 길이의 무한히 많은 시퀀스를 만들어내며, 루프나 분기 등에 의해 학습에 이용된 패턴 이외에 새로운 형태의 행위가 나타날 수 있는 등의 많은 장점이 있기 때문에 지금껏 많이 연구되어왔다[3].

그러나, finite automata는 간결한 형태의 finite automata를 학습하는 것이 매우 어려워, 대부분 이전의 연구에서 제시한 오토마타는 사용자의 개입이 상당 부분 작용했었다. 이후, finite automata와 유사한 HMM을 이용한 연구가 진행되었으나 학습에 지나치게 큰 오버헤드가 있으며, Forrest가 제안한 알고리즘에 비해 비교적 적은 탐지율의 향상을 가져왔다[3].

본 논문에서는 침입 탐지에 이용 할 수 있도록 시스템 콜로부터 정상 행위를 모델링하고, 기존의 finite automata를 통한 모델링 기법들의 단점을 극복할 수 있는 자동화된 finite automata를 생성하는 알고리즘을 제시하고자 한다.

2. 관련연구

2.1 System Call Tracing

Forrest의 연구에서는 실행되는 프로세스의 시스템 콜의 짧은 시퀀스를 정상 행위로 정의하였다. 시스템 콜의 순서적인 정보를 데이터베이스에 입력하고 이후 프로세스를 모니터링 할 때, 프로세스의 시스템 콜이 데이터 베이스에 정의된 시스템 콜의 순서에 맞지 않는 경우를 세어 전체 시스템 콜에 대한 비율로 침입을 판별하였다[4].

그 밖에도 시스템 콜을 데이터 소스로 하는 기법으로는 빈도에 기반 하는 탐지 방법, 데이터 마이닝 기법, finite state machine 등이 연구되었다[5].

2.2 Finite Automata를 이용한 침입 탐지 기법

[6]에서는 각 프로세스에 자주 반복되는 시스템 콜의 묶음을 매크로로 단축하고 전체 길이를 줄인 후에 프로세스들의 시퀀스를 하나의 finite automata로 표현한다. 그러나, 여기서 매크로를 정의하는 부분이나 finite automata를 생성하는 부분이 자동적인 생성이 아니기 때문에 대량의 데이터를 분석하여 많은 정상 행위의 패턴을 프로파일링 하기 어렵다.

[7]은 시스템 콜이 아닌 프로그램의 소스 코드로부터 함수의 콜 그래프 모델로 transition diagram을 생성하였다. 그러나, 프로그램의 소스를 기반으로 설계되는 만큼 다양한 프로세스들을 포함하는 정상 행위를 모델링하기에 어렵다.

[8]에서는 트레이닝 데이터가 많다 하더라도 정상 프로그램의 행위로부터 자동적으로 생성된 finite automata와 통계적인 수치를 이용하여 침입을 탐지하는 방식을 제안하였다.

3. 제안된 프로세스 모델링 기법

본 논문에서 제안하는 기법에서는 프로세스의 정상 행위를 finite automata를 통해 모델링 한다. 다시 말해서 정상적인 시스템 콜 시퀀스들의 집합을 약 190개의 시스템 콜을 알파벳으로 가지는 일종의 formal language로 간주한다는 것이다.

제안하는 방식은 [그림1]에서 보는 바와 같이 긴 시스템 콜의 시퀀스를 슬라이딩 윈도우를 이용해 겹쳐진 형태로 동일한 길이의 짧은 시퀀스로 나누는 분할 단계, 기본적인 오토마타 생성 알고리즘인 전처리 단계, 그리고, 언어압축 알고리즘을 이용한 일반화 단계를 거쳐 우리들이 원하는 정상 행위를 표현하는 오토마타를 만들어낸다.

3.1 전처리 단계

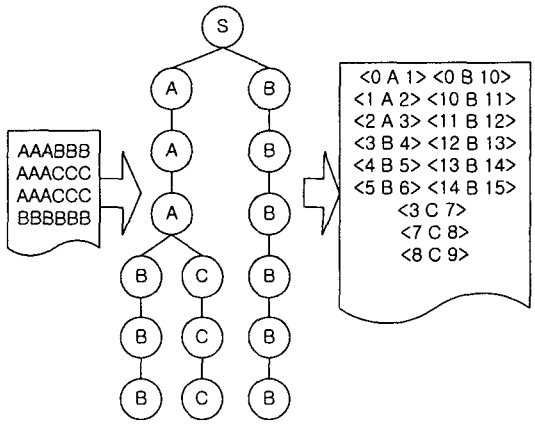

그림 2. 전처리 단계

동일한 길이로 분할된 시스템 콜 시퀀스를 오토마타로 변환하기 위해 트리를 생성한다
이 트리는 입력된 문자열에서 하나의 문자를 스캐닝 하면서 트리의 노드와 비교하여 동일한 알파벳이 존재하지 않을 경우 현재 깊이에서 새로운 노드를 추가하고, 그렇지 않을 경우 동일한 알파벳을 레이블로 가지는 노드로 이동한 후 다시 다음 문자를 스캐닝 하는 과정을 모든 문자열을 트리에 포함시킬 때 까지 계속한다. 이렇게 만들어진 트리는 trie[9]의 구조를 따른다.

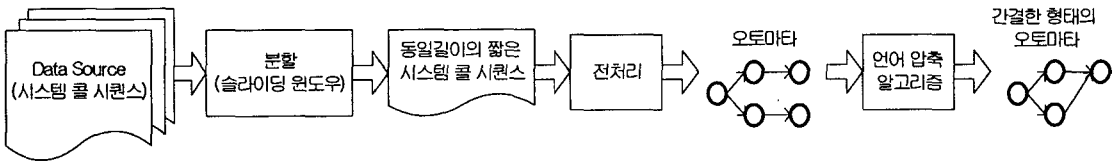

그림 1. FA 생성과정

이렇게 트리를 만들고 나면, 트리를 한번 너비 우선 탐색을 하면서 각 노드의 레이블을 오토마타의 에지의 레이블로 간주하여 오토마타를 생성할 수 있다.

이 전처리 과정을 거치는 동안 동일한 시스템 콜 시퀀스를 제거 할 수 있고, 시퀀스의 앞부분 일부가 동일한 경우에 많은 상태의 수를 줄일 수 있다.

3.2 언어 압축 알고리즘

시스템 콜 시퀀스가 전처리 단계를 지나면, 하나의 오토마타가 생성된다. 이 오토마타는 최소한의 처리만이 거쳐져 거의 트레이닝에 이용된 데이터를 그대로 표현하는 단순한 오토마타이다. 따라서 이 오토마타는 트레이닝에 포함되지 않은 패턴은 100% 모두 비정상으로 판단하게 된다. 그러므로, 이 오토마타에서 실제로 정상 행위의 패턴이라고 여겨질 수 있도록 하기위해서 일반화 단계를 거쳐야만 한다. 그래서, 본 논문에서는 [10]에서 제안한 언어 압축 알고리즘을 이용하고자 한다. 이 알고리즘은 단순히 오토마타의 크기를 줄이는 것에서 그치는 것이 아니라 최초 트레이닝 된 문자열 이외의 새로운 문자열이 인식된다는 점에서도 좋은 선택이 될 수 있다.

언어 압축 알고리즘은 MDL(Minimum Description Length)이라는 개념을 이용한다. DL(Description Length)은 데이터 인코딩 길이(data-encoding length) Δ와 그래마 인코딩 길이(grammar encoding length) Γ의 두 값의 합으로 결정되며, m이 인코딩 되는 문장의 수, $|S_i|$는 i번째 문자열 S_i의 길이, Z_{ij}는 문장 S_i의 j번째 심볼에 도달한 상태를 떠나는 에지의 수, $|\delta|$는 δ에 속하는 triple의 수, $|Q|$는 상태의 수, $|\Sigma|$는 알파벳의 크기, 그리고, $|F|$가 최종 상태의 수라고 할 때, 두 값 Δ와 Γ는 다음과 같이 정의 된다.

$$\Delta = \sum_{i=1}^{m} \sum_{j=1}^{|S_i|} \log_2 z_{ij}$$

$$\Gamma = |\delta|[2(\log_2|Q|) + \log_2|\Sigma|] + |F|[\log_2|Q|]$$

주어진 문자열의 집합으로부터 MDL을 가지는 DFA를 구성하는 알고리즘은 다음과 같다[9].

제 1단계. 입력 문장을 생성한다.
제 2단계. 입력 문장의 집합을 인식하는 DFA를 생성한다.
제 3단계. DFA에서 2개의 상태를 합쳐서 1개의 상태로 만들더라도 NFA가 되지 않고 DFA로 남아있도록 하는 것 중에서 DL을 최소로 하는 2개의 상태를 합쳐서 1개의 상태로 만든다.
제 4단계. 제 3단계를 반복한다. 제 3 단계의 조건을 만족하는 2개의 상태가 존재하지 않으면 종료한다. (hill-climbing)

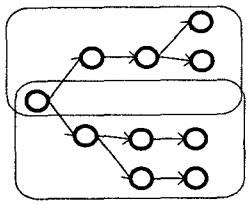

그림 3

그러나, 이와 같은 언어 압축 알고리즘은 계산 복잡도가 높기 때문에 입력으로 주어지는 오토마타의 상태 수가 많을 경우에는 한꺼번에 처리하는데 지나치게 많은 시간을 소요한다. 그래서, 제안하는 시스템에서는 전처리 과정을 거쳐서 생성된 오토마타를 [그림3]과같이 언어 압축 알고리즘에 부담이 되지 않을 정도의 상태 수를 가지도록 분리한 후, 분리된 오토마타 각각에 대해 언어압축 알고리즘을 적용하였다.

4. 실험 및 분석

실험에는 SunOS 4.11이 동작하는 Sun SPARCstation에서 수집된 sendmail 데이터를 사용하였다. 이 데이터는 뉴 멕시코 대학에서 만들어진 것으로, [4]에서 제공한 프로그램을 통해 수집된 시스템 콜이다.

정상 행위를 모델링 하기 위해서 전체 sendmail데이터의 19520개의 시스템 콜 중에서 연속된 70개의 시퀀스를 임의로 네 부분에서 추출하여 프로파일링에 사용하였고, 오토마타가 생성된 후, 정상행위로써 sendmail 데이터의 전체 시퀀스와 비 정상 행위로써 syslog를 이용한 공격을 오토마타에 입력으로 주었다. 이 때, 슬라이딩 윈도우의 크기는 6으로 주고 실험하였다.

표 1. Sendmail 데이터의 테스트

분류 / 데이터		전체 시스템 콜의 수	압축 알고리즘 비 사용	압축 알고리즘 사용
			# of Accept (인식률%)	# of Accept (인식률%)
정상	sendmail	19520	14406(73)	15134(78)
비정상	syslog-Local1	1510	833(55)	885((59)
	syslog-Local2	1568	835(53)	880(56)
	syslog-Remote1	1855	916(49)	1005(54)
	syslog-Remote2	1547	753(49)	843(54)

(인식률: # of Accept / 전체 시스템 콜의 수)

언어 압축 알고리즘을 사용하지 않고 전처리 단계만을 거친 오토마타에 정상 행위를 테스트 한 결과, 트레이닝에 사용된 데이터가 극히 일부에서 추출한 데이터임에도 불구하고, 약 73%를

인식했으며, 여기에 압축 알고리즘을 사용한 경우 약 78%를 인식했다.

그에 반해, 비정상 행위의 데이터를 오토마타에 적용해 본 결과 대부분의 경우에 인식률이 50~60% 대에 머물렀다.

실험 결과에서 시스템 콜을 통해 오토마타로 정상 행위의 패턴을 생성했을 때, 정상 행위와 비정상 행위 사이의 인식률은 확연한 차이를 보였으며, 언어 압축알고리즘을 이용하면 약 5% 이상 인식률이 높아졌다.

비록 5% 정도에 불과하지만, 트레이닝 데이터의 수가 증가하는 것에 비해 인식률 증가의 기울기가 작아지기 때문에, 트레이닝 데이터를 늘리는 것으로 더 이상 인식률 성장의 큰 효과가 없게 되는 경우에 이 5% 가 주는 의미는 매우 크다고 할 수 있다.

5. 결론

본 논문에서 정상 행위의 프로파일링을 위해서 finite automata를 이용하였다.

정상 행위를 위한 finite automata를 생성하기 위해서 우리는 슬라이딩 윈도우를 통해 동일 길이로 잘려진 시스템 콜의 시퀀스를 간단한 알고리즘을 통해 오토마타를 생성하고, 패턴의 일반화를 위해 언어 압축 알고리즘을 적용하였다.

실험 결과, 정상 행위와 비정상 행위 사이에 finite automata의 인식률의 차이를 볼 수 있었고, 침입 탐지를 위한 정상 행위 모델링의 가능성이 충분히 있음을 보여주었다.

그러나, 적은 일부의 데이터 만으로 정상행위를 모델링 하는 것은 가능하지만, 결과적으로 다른 탐지 기법들에 대해 부정적, 긍정적 결함이 매우 높아 질 위험이 존재한다. 또한, 문자열 형태의 패턴을 가지고 공격을 탐지하는 경우에는 정상 행위가 포함하고 있는 패턴만으로 이루어진 공격에 대해서는 탐지 할 수 없다. 예를 들어, 프로그램 안에 무한 루프에 빠진다든지 하는 형태는 서비스 거부 공격의 일종이지만, 짧은 길이의 패턴만으로는 이것을 탐지 하기 어렵다

그러므로, 제안된 기법이 독립적으로 사용되기 보다는 모델링 하거나 탐지하는 부분의 오버헤드가 매우 적다는 장점을 살려 오버헤드가 높은 다른 탐지 기법에 대한 보완책으로 사용할 수 있을 것이다.

참고문헌

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

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

[3] R. Sekar and M. Bendre, "A Fast Automaton-Based Method for 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] C. Warrender and S. Forrest, "Detecting Intrusions Using System Calls: Alternative Data Models", Proceeding of the 1999 IEEE Symposium on Security and Privacy, pp. 133-145, 1999.

[6] A. KIosoresow, "Intrusion Detection via System Call Traces", IEEE Software '97, pp. 35-42, 1997.

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

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

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

[10] T. Teal and C. Taylor, "Effect of Compression on Language Evolution", Artificial Life 6(2), pp. 129-144, 2000.