Event Sequence Tracking을 이용한 침입 감지 시스템의 설계

최 송 관[O], 이 필 중

포항공과대학교 정보통신대학원[*], 포항공과대학교 전자전기공학과

Design of Intrusion Detection System Using Event Sequence Tracking

SongKwan Choe, Pil Joong Lee

Graduate School for Information Tech., Dept. of Eletronics and Electrical Eng. POSTECH

요약

본 논문에서는 컴퓨터 시스템에서 침입 감지 시스템을 설계함에 있어서 사용될 수 있는 새로운 방법인 Event Sequence Tracking 방법을 제안하였다. Event Sequence Tracking 방법에서는 컴퓨터 시스템의 공격 방법을 크게 두가지로 분류한다. 첫번째는 일련의 시스템 명령어를 이용한 공격방법이고 두번째는 침입자 자신이 만들었거나 다른 사람으로부터 얻은 프로그램을 이용하는 방법이다. 첫번째 공격방법에 대한 감지 방법은 시스템을 공격할 때 사용한 일련의 시스템 명령어들을 감사 데이타를 분석하여 찾아내고 이 결과를 기존에 알려진 공격 시나리오들과 비교하여 침입자를 찾아내는 방식이다. 두번째 공격방법에 대한 감지 방법은 보안 관리자가 정해놓은, 시스템에서 일반 사용자가 할 수 없는 행위에 관한 보안 정책에 따라 Key-Event 데이타 베이스를 만들고 여기에 해당하는 event의 집합을 감사 데이타에서 찾아내는 방법이다. Event Sequence Tracking 방법은 Rule-based Penetration Identification 방법의 일종으로서 시스템의 공격방법을 분류하여 컴퓨터 시스템에의 침입을 효과적으로 감지할 수 있다는 것과 rule-base의 생성과 갱신을 함에 있어서 보다 간단하게 할 수 있다는 장점을 갖는다.

1 연구 배경

사회의 발전이 점진적으로 정보에 의존하게 되고 정보의 자동적인 처리방법이 신 기술로 대두되면서 정보를 처리하는 컴퓨터에 정보보호기능의 필요성이 증가하게 되었다. 또 이러한 컴퓨터들이 방대한 정보의 신속한 교환을 위해 네트워크를 구성하게 되면서 정보의 노출위험성이 더욱 커지게 되었다.

최근에와서 국외에서 뿐만이 아니라 국내에서도 네트워크를 통해서 컴퓨터 시스템에 침입하여 정보를 훔치거나 시스템을 파괴하는 일명 해커들이 문제가 되고있다. 그 대표적인 사례가 바로 미국에서 1988년에 발생한 Internet Worm사건이다. 이 사건은 미국의 코넬대학의 대학원생이었던 Robert T. Morris가 제작한 worm 프로그램이 하룻밤사이에 7,000여대의 컴퓨터를 사용불능상태로 만들었던 유명한 사건이다. 그 외에도 1989년에 서독의 해커들이 소련 KGB의 사주를 받아 전세계 200여개의 컴퓨터에 침입하여 주요 군사정보를 탈취하였던 사건, 뉴욕의 414번지에 사는 해커들이 뉴욕 암치료센터의 정보를 지워버렸던 사건등을 포함하여 이미 널리 알려진 사건만해도 매우 많다 [1].

국내에서도 1994년에 영국의 한 소년이 원자력 연구소의 컴퓨터에 침입하였던 사건, 1993년 서울대학교의 전산센타의 6대의 컴퓨터의 hard disk가 지워졌던 사건등 많은 사례가 있다 [1]. 컴퓨터가 해킹을 당하면 대부분의 사람들은 이를 대외적으로 밝히길 꺼려하는 경향이 있다. 사고를 당했다는 사실은 컴퓨터의 보안관리가 미비하다는 것을 말하고 또 이러한 사실을 들은 해커가 그 시스템을 공격할 우려가 있기 때문이다. 실제로 아무런 흔적을 남기지 않고 정보를 가져가는 경우도 있기 때문에 자신의 컴퓨터가 해킹을 당했다는 사실조차 모르게되는 경우도 허다하다. 국내에서 이미 알려진 사건들을 보면 대부분의 경우에 있어서 초보적인 해킹테크닉만으로도 쉽게 시스템에 침입이 가능했다는 사실을 알 수 있다. 이는 국내에서 컴퓨터 보안에 대한 인식이 매우 부족하다는 것을 간접적으로 말해주고 있는 것이다.

컴퓨터 시스템에의 침입은 외부로부터의 침입뿐만이 아니라 적법한 권한을 가진 내부의 사용자에 의한 경우도 매우 많다. 오히려 시스템이 관리하고 있는 정보를 몰래 훔쳐서 외부로 팔아 넘기는 것이 외부에서 훔치려는 노력을 하는 것보다 훨씬 쉽기 때문에 외부의 침입자에 의한 사고 보다 내부의 시스템 사용자에 의한 사고가 더 많은 비율을 차지하고 있다. 미국공군의 전산센터의 보고에 의하면 조사된 컴퓨터 보안사고의 80%가 내부의 시스템 사용자에 의한 사고였다고 한다 [2]. 따라서 시스템을 보호하려는 노력은 외부로부터의 침입자뿐만이 아니라 내부의 시스템 사용자에 대해서도 반드시 이루어져야한다.

최근 컴퓨터 시스템의 보호를 위해서 시스템이 제공하는 감사 데이타(Audit data)를 이용하여 보안사고를 실시간에 발견하거나 보안사고 발생의 원인 규명을 하기위한 도구인 이른바 침입 감지 시스템(Intrusion Detection System)에 관한 연구가 활발하게 이루어지고 있다. 침입 감지 시스템은 매우 많은 양의 감사 데이타를 분석하여 시스템에 관한 보안 위배 사항(Security violation)을 찾아내거나 시스템 사용자들의 행동을 자세히 관찰하여 여러가지 방법을 이용하여 침입자인지 아닌지를 판단한 후 이를 보안 관리자에게 보고하는 기능을 갖고 있다.

2 침입 감지 시스템 (IDS : Intrusion Detection Systems)

침입이란 컴퓨터 시스템의 자원들에 허가받지 않고 접근하거나 그들을 허가받지 않고 사용하는 행위를 말한다. 침입 감지 시스템은 이러한 침입이 발생함을 감지하고 이를 보안 관리자에게 알리거나 그 외의 적절한 대비를 자동적으로 해주는 일을 한다. 이러한 침입 감지 시스템은 일반적으로 컴퓨터 시스템의 운용체제가 제공하는 감사기능을 이용하여 만들어진 감사 데이타를 이용한다. 침입 감지 시스템의 기본적인 메카니즘들은 다음과 같이 크게 두가지로 나누어 볼 수 있다 [3].

1. Intrusion detection expert system
   이미 알려져 있는 침입유형들을 데이타베이스에 모아두고 이들과 시스템의 감사 데이타에서 관찰된 시스템 사용자의 사용패턴을 비교하여 이것이 침입유형과 같으면 침입으로 간주한다.

2. Profile-based intrusion detection system
   시스템 사용자의 일상적인 사용패턴을 이용하여 사용자 profile을 만들어서 감사 데이타에서 관찰한 사용패턴을 profile의 그것과 비교했을 때 다른점이 많으면 침입이라고 간주한다.

위의 두가지 방법은 모두가 컴퓨터 시스템에서 제공하는 감사 데이타의 분석을 기초로 하여 이루어진다. Disk의 용량이 한정되어있는 이상 감사 데이타의 양도 한정되어있다. 위의 두가지 메카니즘은 감사 데이타

의 양이 많을수록 정확한 결과를 제공하므로 가능한한 많은 양의 감사 데이타를 얻는것이 요구된다. 이 문제를 해결하기 위해서 실제로 침입 감지 시스템에서는 감사 데이타의 감소(reduction) 또는 여과(filtering) 기법을 통하여 많은 양의 데이타를 효과적으로 처리한다.

침입 감지 시스템에 관한 연구를 일찌기 수행했던 J.P. Anderson은 감사 데이타를 이용하여 컴퓨터 시스템 사용자의 독특한 사용패턴을 알 수 있다고 했다 [4]. 또, 그는 감사 데이타를 분석함으써 찾아낼 수 있는 침입의 종류를 다음과 같이 분류했다.

1. *External Penetration* : 허가받지 못한 외부인에 의한 침투
2. *Internal Penetration* : 허가받은 내부인에 의한 침투
   (a) *Misfeasor* : 적법한 사용자로서 부당한 행위를 하는 사람
   (b) *Masquerader* : 시스템의 접근제어 기능을 피하여 자기가 아닌 다른 적법한 사람의 계정에 침투하는 사람
   (c) *Clandestine* : 시스템의 감독권한을 갖고 이것을 이용하여 시스템의 감사 기능을 피하여 행동하거나 그것이 제기능을 발휘하지 못하게 만든 후 행동하는 사람
3. *Virus & Trojan horse*
   (a) *Virus* : 실행가능한 프로그램의 부트 섹터에 존재하여 그 프로그램이 실행될 때 그 프로그램에 유해한 행동을 하고 자신을 다른 프로그램에 복사하는 능력을 가진 프로그램을 말한다. 이것은 자신이 스스로 동작하는 프로그램이 아니라는 점에서 worm과는 구별이 된다.
   (b) *Trojan horse* : 시스템을 공격하기 위해 만들어진 프로그램이 아닌 프로그램에 삽입되어 실행 과정에서 사용자가 모르는 사이에 시스템에 유해한 행동을 하는 프로그램을 말한다.

일반적으로 misfeasor를 감지하기가 masquerader를 감지하기보다 훨씬 어렵다고한다. 그것은 misfeasor에 있어서는 정상적인 행동과 비 정상적인 행동을 구별하기가 매우 힘들기 때문이다.

3 침입 감지 메카니즘

3.1 Statistical Anomaly Detection

- Threshold Detection
  어떤 특정한 event를 정해진 시간동안 관찰했을 때 그 event의 발생 빈도수가 평소와는 달리 매우 높다면 침입에 의한 event의 발생이라고 의심할 수 있을것이다. 가령 worm 프로그램의 경우 시스템에 침입한 후에 짧은 시간동안 수 많은 프로세스를 생성시켜서 시스템에 부담을 주어 다른 작업을 할 수 없게 만든다. 이 때에 시스템에서 생성되는 프로세스의 수는 분명히 worm 프로그램이 실행되고 있는 시간에 급격히 증가할 것이므로 그 프로그램은 침입에 의한 프로그램이라고 의심을 받아야한다. 여기에서 문제가 되는 점은 어느 시점부터 의심을 하는가의 문제이다. 만약 어떤 event가 5초동안 100번 이상 발생할 경우 침입이라고 판단한다면 그 100이라는 값이 그 event에 해당하는 threshold인 것

Rule NO.	1	2	3	4	5
1	A	B	C	D	E
2	A	B	D	E	F
3	B	C	F		

표 1: 사용자 '갑'의 Rule-base

이다. 따라서 threshold를 결정하는 문제가 가장 핵심적인 문제이고 이때 고려해야할 사항이 매우 다양하고 복잡하기 때문에 가장 어려운 문제이기도하다. 일반적으로 침입 감지 시스템을 구현하는데 있어서 이 방법은 가장 단순한 메카니즘에 속한다.

- Profile-based
  감사 데이타로부터 시스템 사용자들의 평상시의 사용패턴을 정해진 규칙에 따라 기술하여 profile을 만든다. 그리고 현재 시스템 사용자의 행동을 감사 데이타에서 읽은 후 이것을 profile과 비교하여 통계적인 방법으로 분석을 한다. 그 분석한 결과에 의해 현재의 사용패턴이 profile과 비교해 볼 때 많이 다르다고 판단되면 침입이라고 간주한다. 이 방법의 큰 장점은 보호하려는 컴퓨터 시스템이 갖고 있는 보안상의 미비점들(security flaws)을 미리 고려할 필요가 없기 때문에 시스템의 종류와 무관하게 어떤 시스템에서라도 사용될 수 있다는 점이다. 그리고 이 방법으로 구현된 S/W는 그대로 다른 메카니즘으로 구현된 S/W와 함께 침입 감지 시스템을 구현하는데 사용할 수 있다.

3.2 Rule-based Anomaly Detection

Statistical Anomaly Detection과 비교해 볼 때 가장 큰 차이점은 Statistical Anomaly Detection은 profile을 만들 때 통계적인 수치를 이용하는 반면에 Rule-based Anomaly Detection에서는 몇 가지 규칙을 이용한다는 점이다. 여기서 말하는 규칙은 과거에 만들어진 감사 데이타로부터 얻은 시스템 사용자의 사용패턴을 말한다. 보통 이러한 규칙들은 rule-base에 저장되어 있어서 새로이 얻어진 사용패턴과 비교하는데 사용된다. 예를 들면, 갑이라는 사용자는 (표 1)과 같은 rule-base를 갖는다고 가정해보자. 현재 감사 데이타로부터 "A $\longrightarrow$ B $\longrightarrow$ F" 라는 사용패턴을 얻었다면 이러한 행동은 침입으로 간주된다. 왜냐하면 rule-base에서 보면 B의 다음에는 C나 D만이 따라와야되기 때문이다.

3.3 Rule-based Penetration Identification

기본적으로는 Rule-based Anomaly Detection과 같다고 할 수 있다. 이미 알려져 있거나 보안 관리자에 의해 분석된 침입유형을 일정한 형태로 만들어서 이를 rule-base에 저장한다. 그래서 이러한 침입유형과 감사 데이타에서 관찰된 행동을 비교한 후 같은 것만을 침입이라고 간주한다. 즉, Rule-based Anomaly Detection에서는 일반적인 사용패턴을 rule-base에 저장하여 이것과 다른 사용패턴을 침입이라고 간주하지만 Rule-based Penetration Identification에서는 시스템의 침입자가 사용하는 침입유형을 rule-base에 저장하여 이와 같은 사용패턴을 침입으로 간주하는것이다.

3.4 State Transition Analysis

이 방법에서는 시스템 내부에서 침입자가 시스템의 관리자 권한을 얻기까지의 과정을 단계별로 기술하여 knowledge-base에 저장한 다음 이를 감사 데이타에 나타난 사용자의 사용패턴과 비교하여 침입자를 찾아낸다. 여기서 침입자가 시스템의 관리자 권한을 얻는데 사용하는 침입방법을 knowledge-base에 기술하기 위해 사용하는 방법을 State Transition Analysis라고 한다. 즉, 하나의 공격 시나리오에서 이 공격방법이 성공적으로 이루어지기 위해 반드시 필요한 행동들과 그러한 각각의 행동들로 인해 새로이 생긴 시스템의 상태를 규명하는 방식을 State Transition Analysis라고 한다 [6, 7].

4 Event Sequence Tracking을 이용한 침입 감지 시스템

4.1 C2 보안등급 감사 레코드

NCSC (Nationcal Computer Security Center)에서 정의한 컴퓨터의 보안등급을 설명하면 다음과 같다 [9].

- D 등급 : 아무런 보안장치가 달려있지 않은 개인용 컴퓨터에서와 같이 특별한 보안 장치가 없는 상태를 말한다. 보안등급중 가장 낮은 등급이다.
- C1 등급 : 보통의 UNIX 시스템에서와 같이 사용자가 시스템을 사용하려면 login name과 password를 입력해야하며 사용자를 user, group, others의 계층으로 분류한다.
- C2 등급 : 보안에 관계되는 event들이 감사되고 login name과 password를 입력함으로써 인증을 한다. 보안 옵션이 있는 SunOS가 대표적인 예이다.
- B1 등급 : 시스템 내부의 주체의 신뢰도에 따라 보안 레이블을 부여하여 이 주체가 어떤 객체에 접근할 때 보안 레이블에 따라 접근 여부가 결정된다.
- B2 등급 : 시스템의 보안 상태가 완전히 규명되어야하고 조절될 수 있어야한다. 시스템의 관리자, 보안 관리자, 오퍼레이터의 기능이 명확히 구별되어야하고 security hole이 전혀 없어야한다.
- B3 등급 : Access contol list에 의해서 접근제어가 이루어 지고 시스템의 내부구조가 완전히 규명되어야한다.
- A1 등급 : 시스템의 보안 상태가 안전하다고 증명되어야한다.

본 논문에서는 보안 등급이 C2인 SunOS 4.1.3을 이용하여 감사 데이타를 생성하여 이것을 자료로하여 예를 들고자한다.

4.2 감사 레코드의 분석

생성된 감사 레코드중의 하나를 보면 다음과같다.

,open,data_read,Sun Aug 27 03:34:38 1995,nolja,nolja,nolja,30 ,290,0,3,(null),30,/
,/robin/user/nolja,/etc/termcap,

여기에서 ','는 field 구분자이다.

감사 레코드는 11개의 header field와 몇 개의 data field로 구성되며 위의 감사 레코드를 각 field별로 나타내면 다음과 같다.

- Record type : open (system call)
- Record event : data read
- Time : Aug 27 03:34:38 1995
- Real user id : nolja
- Audit user id : nolja
- Effective user id : nolja
- Real group id : 30
- Process id : 290
- Error code : 0
- Return value : 3
- Label : null
- data field
  - current root : /
  - current working directory : /robin/user/nolja
  - file name : /etc/termcap

위에서 data field는 사용된 system call의 종류에 따라서 항목의 수와 내용이 달라진다.

4.3 Event Sequence Tracking 메카니즘

감사 데이타에 나타나는 감사 레코드의 나열은 컴퓨터 시스템 사용자들이 실행시킨 명령어들을 시간에 따라 배열해 놓은 것이라고 할 수 있다. 하나의 명령어는 대부분이 두개 이상의 감사 레코드를 만들고 하나의 감사 레코드는 하나의 event에 의해 생성된다. 여기서 말하는 event란 시스템 명령어의 동작을 수행하는데 필요한 기본적인 행위(unit action)를 말하며 감사 레코드의 header field중에서 record type이 이에 해당한다. 다음 페이지의 (표 2)는 UNIX(SunOS 4.1.3) 시스템에서 자주 사용하는 시스템 명령어들

중 'ls', 'ps', 'cd'를 실행시켰을 때 얻어진 감사 데이타를 정리한것들이다. 여기에서 Event-Object쌍의 시간에 따른 배열을 ES(Event Sequence)라 부르기로 한다. Sequence #1은 시스템 명령어의 호출에 해당하는 감사 레코드이고 그 외의 과정은 호출된 명령어가 시스템 내부에서 어떤 행위를 하는지를 서술하고 있다. (표 2)에서 보면 각각의 명령어에 대한 ES가 모두 다름을 알 수 있다. 즉, 하나의 시스템 명령어는 그것에 해당하는 유일한 ES를 갖는다. [1]

시스템에서 사용자가 행하는 작업들을 다음과 같이 요약해 볼 수 있다

- 시스템 명령어를 사용하는 작업(예:'ls -l', 'cp')
- 텍스트 file을 편집하는 작업(예: 'vi'를 이용한 편집 작업)
- 자신이 만든 프로그램을 실행하는 작업(예: 'a.out')
- 응용 프로그램을 실행하는 작업(예: 'openwin', 'startx')

이러한 작업들은 시스템의 입장에서 보면 모두가 시스템에서 제공하는 실행 file을 연속적으로 실행시키는 것에 지나지 않는다. 이러한 연속적인 실행은 event의 단위로 감사 데이타에 기록된다. 결국 시스템 사용자가 구체적으로 어떤 명령어를 사용하였는지 또는 사용자가 실행 시킨 프로그램은 어떤 일을 하는지는 감사 데이타를 분석함으로써 모두 밝혀낼 수가 있다.

(표 2)에서 보면 'ls', 'ls -l', 'ps'라는 명령어는 모두 Sequence #1의 object field에 실행 file 이름이 기록되어있다. 따라서 ES를 관찰함으로써 시스템 사용자가 어떤 실행 file들을 사용했는지를 알려면 ES에서 Sequence #1의 object field를 보면 알 수 있다고 말할 수 있다. 그런데 시스템의 침입자들은 시스템 명령어를 이용하여 침입을 시도할 경우 일반적으로 시스템 명령어가 갖는 옵션들의 특성을 이용하여 공격을 시도하므로 그들의 행동을 정확히 규명하기 위해서는 어떤 옵션을 사용하였는지도 알 수 있어야한다. ES에서 Sequence #1의 object field에서는 사용한 명령어의 옵션에 대한 정보를 얻을 수가 없으므로 결국은 사용한 옵션을 알기 위해서는 ES의 모든 노드를 관찰하여야한다. (표 2)에서 'ls'와 'ls -l'의 감사 데이타를 비교하면 Sequence #19에서부터 달라지기 시작함을 관찰할 수 있다. 즉, 'ls'와 'ls -l'은 각각의 ES를 비교함으로써 구분할 수 있다. 따라서 모든 시스템 명령어의 ES와 각각의 명령어의 옵션에 대한 ES를 모두 데이타베이스에 저장해놓고 이것과 새로이 생성되는 감사 데이타에서 얻어진 ES를 비교하면 시스템 사용자가 어떤 실행 file을 사용하였으며 어떤 옵션을 주었는지를 정확히 파악할 수 있을것이다.

그런데 'cd'의 Sequence #1은 event가 execve도 아니고 object field에 실행 file의 이름이 기록되어있지도 않다. 그 이유는 'cd'는 그 자체가 'chdir'이라는 하나의 system call로만 구성이 되어있는 명령어이기 때문이다. 따라서 이러한 종류의 명령어는 ES의 Event field만으로도 충분히 파악이 가능하다.

시스템 사용자의 행동들중에서 규명해야할 나머지 한가지는 사용자 자신이 어떤 프로그램을 실행시키는 행동이다. 실제로 해커들의 공격은 단순한 시스템 명령어들의 조합으로 이루어지는 경우보다 시스템에 대한 해박한 지식을 이용한 침입방법을 자신의 프로그램으로 구현하여 이를 실행시킴으로 이루어지는 경우가 많다. 가령 어떤 해커가 하나의 공격 프로그램을 만들어서 실행 file로 만들어 Internet에 공개했다고 가정해보자. 그리고 그 프로그램이 자신이 개발한 새로운 편집기라고 거짓으로 소개했다고 하자. 만약 그러

[1] 똑같은 시스템 명령어라도 OS의 version이나 그것이 사용하고 있는 응용 프로그램에 따라 ES는 다를 수 있다. 그러나 본 논문에서는 예를 들어 설명하기 위해서 그러한 다양성에 대한 고려는 하지않았다.

Command(PID)	Sequence #	Event	Object
ls(261)	1	execve	/bin/ls
	2	open	/usr/lib/ld.so
	3	open	/dev/zero
	4	open	/etc/ld.so.cache
	5	open	/usr/lib/libc.so.101.9
	6	open	/usr/lib/libdl.so.1.0
	7	stat	/lib/locale/korean
	8	open	/lib/locale/korean/LC_CTYPE
	9	stat	/lib/locale/korean
	10	open	/lib/locale/korean/LC_TIME
	11	stat	/lib/locale/korean
	12	open	/lib/locale/korean/LC_NUMERIC
	13	stat	/lib/locale/korean
	14	stat	/lib/locale/korean
	15	open	/lib/locale/korean/LC_MONETARY
	16	stat	.
	17	stat	.
	18	stat	*my directories and files*
ls -l(267)	1 - 18	'ls'의 경우와 동일함	
	19	open	/etc/passwd
	20	socket	0
	21	socket	17
	22	open	/usr/share/lib/zoneinfo/localtime
ps(279)	1	execve	/bin/ps
	2	open	/usr/lib/ld.so
	3	open	/dev/zero
	4	open	/etc/ld.so.cache
	5	open	/usr/lib/libkvm.so.0.3
	6	open	/usr/lib/libc.so.101.9
	7	open	/usr/lib/libdl.so.1.0
	8	stat	/lib/locale/korean
	9	open	/lib/locale/korean/LC_CTYPE
	10	open	/etc/psdatabase
	11	stat	/vmunix
	12	open	/vmunix
	13	open	/dev/kmem
	14	open	/dev/mem
	15	open	/dev/drum
cd(273)	1	chdir	/

표 2: 'ls','ls -l','ps','cd'의 감사 데이타

한 편집기 프로그램을 원하던 사용자가 그 프로그램을 가져다가 아무런 생각없이 그대로 사용한다면 그 프로그램을 실행시킨 시스템은 해커의 침입을 피하기 힘들것이다. 따라서 그러한 프로그램들은 사용하기전에 그 프로그램이 제작자가 명시한 행동 이외의 어떤 위험한 행동을 하지 않는지를 확인할 필요가 있다. 그러한 확인작업을 Key-Event Sequence Tracking 방법을 이용하여 할 수 있다. 이 방법은 Event Sequence Tracking 방법을 이용하여 프로그램이 하는 행동들 중에 위험한 행동을 감지해내기 위해 감사 데이타를 event를 단위로 분석하여 Key-Event 데이타베이스의 내용과 비교하는 방식을 사용한다. Key-Event 데이타베이스에는 일반 사용자들이 접근해서는 안되는 file들의 목록과 관찰하려는 주요한 event의 목록이 기록되어있다. 이 방식을 이용하여 Internet에서 얻은 어떤 프로그램이 믿을만한 것인지를 확인하려면, 먼저 침입을 당해도 무방한 전용 시스템에서 그 프로그램을 작동을 시킨 다음 그것에 대한 감사 데이타를 얻어서 이를 입력으로하여 Key-Event Sequence Tracking 방법을 이용한 침입 감지 시스템을 작동 시켜보는 것이다. 이렇게 하면 그 프로그램이 정확히 무슨 행동을 하는지까지는 알 수 없더라도 위험한 프로그램인지 아닌지는 확인할 수가 있을것이다.

5 비교 및 분석

기존의 대부분의 침입 감지 시스템이 수용하고 있는 방법인 Statistical Anomaly Detection 방법이 갖는 첫번째 취약점은 시스템의 공격자가 침입 감지 시스템을 오랜 기간의 훈련을 통하여 속일 수 있다는 점이다 [10, 11]. 공격자가 인내심을 갖고 아주 조금씩 원래의 행동에서 벗어나는 행동(anomaly)들을 하게 되면 침입 감지 시스템을 훈련 시킬 수가 있다. 그래서 결국에는 어떠한 행동을 해도 침입 감지 시스템이 그것을 감지해 낼 수가 없게된다. 두번째 취약점은 비록 침입자가 행하는 하나하나의 행동이 anomaly가 아니더라도 그들의 조합을 이용하면 시스템을 공격할 수 있는 방법을 많이 만들 수 있다는 점이다. 이러한 문제점들을 해결하기 위한 방법이 Rule-based Penetration Identification 이라고 할 수 있다. 앞에서 살펴본 바에 의하면 Event Sequence Tracking 방법은 침입 감지 시스템의 분류상 Rule-based Penetration Identification에 속하고 State Transition Analysis의 방법도 부분적으로 수용하고있다. 따라서 본 논문에서 제안한 방식인 Event-Sequence Tracking 방법은 위에서 제시한 Statistical Anomaly Detection의 두가지 문제점을 해결할 수 있다.

State Transition Analysis 방법에서는 하나의 침입 시나리오를 rule-base에 기록함에 있어서 보안관리자가 침입 시나리오를 분석하여 침입자의 침입과정에 따른 state의 변화를 일일이 knowledge-base에 기록해야한다는 문제점이 있다. 이에 비해 Event Sequence Tracking 방법은 침입자가 침입과정에서 사용한 일련의 명령어들을 그대로 rule-base에 기록만하면 되기 때문에 침입방법에 관한 깊은 지식을 요구하지 않는다. 예를 들면 UNIX BSD 4.1.1 version에서 sendmail이 동작할 때 입력으로 들어오는 file의 set-uid bit가 reset 되지않음을 이용한 공격은 다음과 같이 이루어진다 [6, 7, 8].

*step*1 cp /bin/sh /usr/spool/mail/root

*step*2 chmod 4755 /usr/spool/mail/root

*step*3 touch x

*step*4 mail < x

*step*5 /usr/spool/mail/root

Step ID	Step	Scenario 1	Scenario 2	Scenario 3
1	cp /bin/sh /usr/spool/mail/root	1	3	1
2	chmod 4755 /usr/spool/mail/root	2	1	3
3	touch x	3	2	2
4	mail < x	4	4	4
5	/usr/spool/mail/root	5	5	5

표 3: Rule-base의 예

위의 공격방법은 rule-base에 (표 3)과 같이 기록될 수 있을것이다. (표 3)에서 Scenario 2, Scenario 3이 생길 수 있는 이유는 step 3이 step 4보다 먼저 이루어지기만하면 공격은 성공할 수 있기 때문이다.

State Transition Analysis 방법에서 침입방법의 분류가 없이 모든 침입을 동일한 방식으로 분석한다는 점도 침입감지의 정확성에 있어서 문제가 된다. Event Sequence Tracking 방법에서는 시스템 명령어의 조합을 이용한 공격과 침입자 자신이 직접 제작한 프로그램을 이용한 공격으로 공격방법의 종류를 분류를 하였다. 그리하여 전자의 경우에는 감사 데이타를 분석하여 침입자가 사용한 일련의 명령어들을 모두 알아내어 이것을 공격 시나리오 데이타베이스의 내용과 비교하는 방법을 사용한다. 이러한 방법을 사용함으로써 (표 3)에서 보는 바와같이 rule-base에 침입 시나리오를 기록함에 있어서 Event Sequence Tracking 방법 자체에 관한 특별한 지식이 없이도 쉽게 기록할 수 있도록 하였다. 후자의 경우는 프로그램이 하는 행동들 중에 위험한 행동을 감지해내기 위해 Key-Event Sequence Tracking 방법을 제안하였다.

Event Sequence Tracking 방법이 갖는 가장 큰 문제점은 시스템 명령어들과 그 밖에 시스템이 사용하는 응용 프로그램들에 대한 Event-Sequence 데이타베이스를 구축하는 문제이다. Event-Sequence의 특성상 사용하는 운용체제나 응용 프로그램에 따라 각각 다른 Event-Sequence 데이타베이스를 구축해야하고 만약 침입 감지 시스템을 실시간으로 작동시키려면 Event-Sequence 데이타베이스에서 찾고자 하는 Event-Sequence를 찾는데 소요되는 시간을 줄이기 위한 효과적인 방법도 연구가 되어야할 것이다.

6 결론 및 향후 연구과제

제안한 Event-Sequence Tracking 방법은 기본적으로는 Rule-based Penetration Identification의 방식을 사용하면서 시스템에의 침입방법을 두가지로 분류하여 침입 감지를 효과적으로 수행하기 위한 방식이다. 따라서 침입 감지 시스템을 구현할 때 기존의 Statistical Anomaly Detection 방법과 함께 구현된다면 보다 효과적으로 침입 감지를 할 수 있으리라 생각된다.

Rule-based Penetration Identification 방식이 갖고 있는 가장 큰 취약점은 알려지지 않은 침입방법에 의한 공격을 감지할 수 없다는 점이다. 현재 이러한 문제를 해결하기 위해서 인공지능 또는 신경망 분야에서도 침입 감지 시스템에 대한 연구가 활발히 진행되고 있다고 알고 있다. 또한 Event Sequence Tracking 방법을 침입 감지 시스템에 적용함에 있어서 연구되어야 할 문제는 알려지지 않은 침입방법에 의해서 시스템이 침입을 당했을 때 그 새로운 침입방법을 분석하여 rule-base를 자동적으로 갱신하는 메카니즘에 관한 연구이다. 그리하여 침입 감지 시스템이 사용되고 있는 동안에도 점차 성능이 향상되도록 만들어야한다.

참고 문헌

[1] 임채호,"인터네트/유닉스 보안", NETSEC-KR'95 Tutorial #1, 1995.6.

[2] Paul Boedges,"Air Force mounts offensive against computer crime", Government News[142].p.51, 12-16th, Orlando, FL.

[3] Mansour Esmaili, Reihaneh Safavi-Naini, Josef Pieprzyk, "Intrusion Detection: A Survey", Submission to ICCC'95.

[4] J.P. Anderson, "Computer Security Threat Monitoring and Surveillance", James P. Anderson's Co., Fort Washington, PA, Apr. 1980.

[5] S. I. Schaen, B. W. McKenny, "Research, Standards and Policy Directions for Network Auditing", 5th Intrusion Detection Workshop, SRI International Menlo Park, California, May 10-11, 1990.

[6] Phillip Andrew Porras,"STAT : A State Transition Analysis Tool For Intrusion Detection", Computer Science Dept., University of California, Santa Barbara, 1992.

[7] Koral Ilgun, "USTAT : Real-Time Intrusion Detection System", Computer Science Dept., University of California, Santa Barbara, Jan. 1992.

[8] Phillip A. Porras, Richard A. Kemmerer, "Penetration State Transition Analysis:A Rule-based Intrusion Detection Approach", *IEEE Computer Society Symposium on Research in Security and Privacy*, 1992.

[9] Sun Microsystems, Inc., SunOS Release 4.1.3 C2 Security Features Guide.

[10] H. S. Vaccro, G. E. Liepins, "Detection of Anomalous Session Activity", *Proceedings of The IEEE Symposium on Research in Security and Privacy*,Okland, pp. 280-289, 1989.

[11] Harold S. Javitz, Alfonso Valdes, "The SRI Statistical Anomaly Detector", *IEEE Computer Society Symposium on Research in Security and Privacy*, 1991.