UNIX 환경에서 퍼지 Petri-net을 이용한 호스트 기반 침입 탐지 모듈 설계

김 민 수[†] · 은 유 진[††] · 노 봉 남[†††]

요 약

본 논문에서는 알려진 침입방법에 대한 탐지 규칙을 생성하는 정형화된 방법을 제안하였다. 그리고, 변형 공격 등의 다양한 공격 방법에 대응할 수 있도록 퍼지(fuzzy)이론을 적용한 퍼지 Petri-net을 제안하였다. 침입탐지에서 이용할 탐지 모듈을 생성할 때, 새로운 패턴이 발견될 때마다 입력하여 새로운 모듈을 추가할 수 있도록 하였다. 또한, 침입 탐지의 정확성을 높이기 위해 시스템 호출 로깅 방법을 사용하였다.

The Design of Host-based Intrusion Detection Module using Fuzzy Petri-net for UNIX

Min-Soo Kim[†] · You-Jin Eun[††] · Bong-Nam Noh[†††]

ABSTRACT

In this paper, we propose formalized method to create detection rules for known intrusion method and the fuzzy Petri-net using fuzzy theory to cope with varied attack. On producing the detection module for using intrusion detection, we can add new found pattern. And also, we use system call logging for increasing correctness of detection.

1. 서 론

침입탐지 시스템은 불법적인 사용, 오용, 또는 불법적인 사용자나 외부 침입자에 의한 컴퓨터 시스템을 남용하는 침입을 알아내려는 시스템이다[2]. 이러한 시스템은 단일 컴퓨터나 네트워크로 연결된 여러 컴퓨터를 감독한다. 침입탐지 시스템은 감사 기록, 시스템 테이블, 네트워크 부하(traffic) 기록의 자원으로부터 사용자 행위에 대한 정보를 분석하여 수행된다.

※ 본 논문은 한국정보보호센터의 용역과제로 수행되는 연구과제임.
† 정 회 원 : 전남대학교 대학원 전산통계학과
†† 준 회 원 : 한국정보보호센터
††† 종신회원 : 전남대학교 전산학과 교수
논문접수 : 1998년 10월 19일, 심사완료 : 1999년 5월 19일

침입탐지 시스템의 목표는 크게 두 가지 방법으로 이루어진다. 하나는 컴퓨터 자원의 비정상적인 행위나 사용을 탐지하는 이상탐지(anomaly detection)이고, 다른 하나는 시스템이나 응용 프로그램의 약점을 통하여 시스템에 침입할 수 있는 잘 정의된 공격을 탐지하는 오용탐지(misuse detection)이다[10]. 침입탐지에 대한 현재 기술적 실제적 논점은 실제 침입이 아닌데 침입으로 판정하는 경우(false positives)와 실제 침입인데 탐지하지 못하는 경우(false negatives)를 어떻게 처리하는가에 있다[15].

본 논문에서는 알려진 침입방법에 대한 탐지 규칙을 생성하는 정형화된 방법을 제안하였다. 이것은 Petri-net을 사용한 방법이 이벤트에 따른 상황을 표현하기 쉽

고 침입상황을 분석하기에 편리하기 때문에 탐지 규칙을 Petri-net 형태로 구성하였다. 보통 규칙기반 침입탐지 방법에서 상태전이 다이아그램의 사용은 트랜지션 분기와 같은 경우를 표현할 수 없으며 이해하기가 복잡하다.

변형 공격 등의 다양한 공격 방법에 대응할 수 있도록 퍼지(fuzzy)이론을 적용한 퍼지 Petri-net을 제안하였다. 기존의 오용탐지 방법에서는 공격형태가 조금 변형된 경우 탐지할 수 없었다. 여기에 본 논문에서는 이상 탐지의 개념을 접목하여 변형 공격에 유동적으로 대처할 수 있도록 하였다. 그것은 침입 패턴에 대한 Petri-net의 각 플레이스에 퍼지값을 두는 것이다.

침입탐지에 이용할 탐지 모듈을 생성하는 프로그램을 작성하였으며 확장성을 제공하기 위하여 새로운 패턴이 발견될 때마다 입력하여 새로운 모듈을 생성할 수 있도록 하였다.

본 논문에서는 침입 탐지의 정확성을 높이기 위해 시스템 호출 로깅을 사용하여 로그 자료를 수집하였다. 기존의 침입탐지 시스템에서 시스템 호출을 이용하는 방법은 많이 있으나, 대부분 파일의 읽기와 쓰기거나 실행(*exec*)의 경우만 처리하였다. 본 논문에서는 침입의 상황을 정확히 표현하기 위해 필요한 모든 시스템 호출을 이용하였다.

2장에서는 로깅방법, 침입탐지 시스템, 침입 방법에 대한 관련연구를 설명하였다. 3장에서는 침입탐지 시스템이 갖는 구조를 설명하였다. 4장에서는 탐지규칙에 적용되는 이론과 수식을 설명하였고, 5장에서 결론을 맺었다.

2. 관련 연구

2.1 로깅(logging) 방법

기존 UNIX 시스템에서 사용하는 로깅 방법을 알아본다. 이러한 로깅에는 사용자가 언제 접속하였는지를 기록하는 내용과 사용자가 무슨 일을 했는지를 기록하는 것으로 나누어 볼 수 있다.

사용자 접속 상황을 기록하는 도구로는 utmp와 wtmp가 있다. 시스템에서 사용자가 사용한 명령어를 기록하는 acct가 있다. 또한, syslog는 전체적으로 syslog에서 지정하는 메시지에 대한 내용을 기록하고 su 명령어 활동을 기록한다. 또한, 모든 sendmail 사용 내역을 기록하여 메일에 관련된 오류를 악용하여 SMTP 포트에 접근하는 것을 알아낼 수 있다.

운영체제에서 제공하는 시스템 호출 로깅 프로그램이 있다. 시스템 호출 로깅은 운영체제마다 다른 프로그램이 존재한다. Solaris 2.x 시리즈의 경우는 BSM (Basic Security Module)이 있다[17]. BSM은 UNIX 시스템 호출을 기록함으로써 어떤 사용자가 어떤 파일을 언제 어떻게 접근했는지를 알 수 있다.

명령어를 로깅하는 방법을 이용하는 침입탐지 시스템에는 몇 가지 문제점이 있다. 첫째, 명령어 이름에만 의존하기 때문에 명령어 이름을 바꾸는 경우에는 다른 명령어로 해석할 수 있다. 둘째, 프로세스가 *fork*나 *exec*를 하는 경우 적절히 대처할 수 없다. 셋째, 프로그램 내에서 사용하는 유해한 시스템 호출을 발견할 수 없다.

따라서, 시스템 호출을 로깅하는 방법이 필요하다. 시스템 호출을 로깅하는 것은 UNIX 커널의 도움이 필요하다. UNIX 명령어로는 trace나 strace가 있는데, 기술한 특정 명령어에 대한 시스템 호출을 기록하거나 지정한 프로세스의 시스템 호출 내역을 기록한다. 그렇지만 trace를 이용하려면 특정 프로세스를 지정해야 하므로, 전체적인 침입 상황을 감시하려면 복잡한 메커니즘이 필요로 하게 된다.

2.2 침입탐지 시스템의 현황

기존의 침입탐지 시스템은 각기 다른 특징을 가지고 있다. 이상과 오용 탐지를 하는가, 어떠한 방법으로 침입탐지를 수행하는가, 탐지 범위는 어디까지 하겠는가에 따라 여러 종류로 나뉘어짐을 <표 1>을 통해서 알 수 있다.

〈표 1〉 침입 탐지 시스템 비교

침입탐지 시스템 종류	탐지 내용		탐지 범위			탐지 방법		
	이상 탐지	오용 탐지	호스트 기반	멀티 호스트 기반	네트워크 기반	통계적	규칙 기반	상태 변이
ASAX[4]		✓		✓			✓	
CMDS[14]	✓	✓		✓		✓	✓	
Computer Watch[3]		✓	✓				✓	
CSM[19]		✓			✓			
DIDS[16]				✓			✓	
EMERALD[13]		✓			✓		✓	
IDES[6,9]	✓		✓			✓		
IDIOT[1]		✓	✓					✓
STAT[5]		✓	✓				✓	✓
W&S[8]	✓		✓			✓		

이러한 시스템에서 탐지하는 내용을 보면 주로 오용을 탐지하는 방법이 많이 사용하고 있음을 알 수 있다. 이상을 탐지하는 방법은 주로 통계적 탐지 방법인데, false positives와 false negatives가 높게 나타난다. 탐지 범위는 호스트 기반에서 다중 호스트 기반 그리고 네트워크 기반까지 다양하고 이러한 방법 세 가지는 항상 유기적인 관계를 가지고 연결되어야 한다. 침입탐지 방법은 오래 전에 나왔던 방법에 통계적인 방법이 많았고, 현재 주로 나오는 방법은 규칙 기반이거나 상태 변이 기반의 시스템이 많다. 본 시스템은 상태 변이를 추적하는 방법으로 침입탐지를 수행한다.

로깅 프로그램과 침입탐지 시스템과의 연결 방법이 필요하다. 호스트 기반과 멀티호스트기반의 침입탐지 시스템에서는 대부분 로깅 프로그램에 의해서 생성된 로그 파일을 이용하는 방법으로 수행된다. 네트워크 기반의 침입탐지 시스템에서는 네트워크 라인 상의 패킷을 분석하여 침입탐지에 필요한 정보를 획득한다. 호스트 기반에서의 침입탐지는 주로 시스템의 접근을 막는 것보다 시스템에서 불법적인 행위를 하는 것을 막는 것으로서 가장 근본적인 침입탐지 방법이다. 본 논문에서는 호스트 기반으로 시스템 호출 기록을 분석하여 상태 변이 방법으로 오용탐지를 수행하는 침입탐지 시스템을 설계하였다.

2.3 침입 방법

다음은 알려진 방법으로 최근에 가장 많이 이용되는 침입형태이다[20,21].

2.3.1 임의의 파일 생성

UNIX 시스템에서 rlogin이 가능한 호스트를 기록하는 hosts.equiv와 .rhosts 파일을 원하는 곳에 생성하거나 링크시키는 방법으로 접근할 수 있다.

또한, /tmp와 같은 누구나 쓸 수 있는 곳에 관리자 권한의 임의 파일을 생성하여 해킹에 이용할 수 있다.

2.3.2 버퍼 오버플로우(buffer overflow)

UNIX내의 프로그램을 수행하는데 명령어와 매개변수를 전달할 때, 매개변수 뒤에 다른 코드를 집어넣어 버퍼 오버플로우를 유발시킨다. 이렇게 오버플로우가 발생했을 때 쉘 프로그램이 실행되도록 하여 관리자 권한을 얻을 수 있다.

2.3.3 서비스 방해 공격

네트워크를 통하여 대량의 메일을 보내거나 쓸모 없는 문자를 계속 목적 호스트에 보내어 시스템의 작동을 방해하는 공격이다. 이 방법은 최근에 늘어나는 공격방법으로 방화벽(firewall)이나 패킷 필터링을 통하여 대응할 수 있다.

2.3.4 스니핑(sniffing)

스니핑은 네트워크의 한 호스트에서 실행되어 그 주위를 지나다니는 패킷을 엿보는 방법으로 ID와 패스워드를 알아내기 위하여 침입자들에 의해 자주 사용된다. 네트워크 보안에 신경을 쓴 호스트라도 주변의 호스트가 공격당해서 스니핑을 위해 사용된다면 무력해질 수밖에 없다.

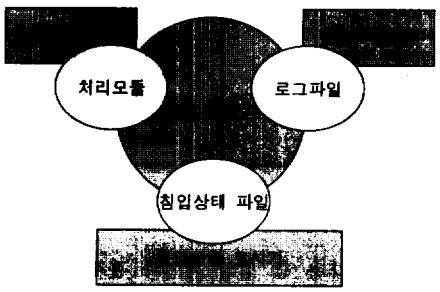
2.3.5 스푸핑(spoofing)

스푸핑이란 자신을 타인이나 다른 시스템에게 속이는 행위를 의미한다. 예를 들어, 특정 호스트에게만 접근권한을 준다고 가정했을 경우 해커는 당연히 자신이 특정 호스트로부터 접근하려는 것처럼 속이려 할 것이며, 이를 가리켜 바로 스푸핑이라고 할 수 있는 것이다.

호스트 기반 침입탐지 시스템에서 탐지할 수 있는 것은 임의의 파일 생성 공격방법과 버퍼 오버플로우 공격방법이다. 따라서, 본 논문에서도 이 두 가지 유형의 공격을 탐지하도록 하고 있다.

3. 침입탐지 시스템 구조

전체적인 모듈간의 관계는 (그림 1)과 같다. 로깅시스템, 침입탐지기, 침입 상황 표시기는 실시간 탐지를 위하여 시스템에서 함께 동작된다. 로깅 시스템에서는

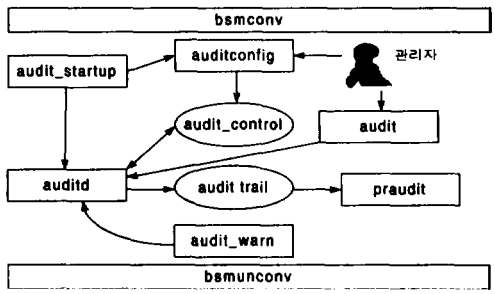


(그림 1) 침입탐지시스템 구조

침입탐지에 사용할 로그 자료를 수집한다. 탐지 모듈 생성기는 새로운 침입 패턴을 추가하는 작업을 수행한다. 탐지 모듈 생성기에서 생성된 처리 모듈에 로그 자료를 적용하여 침입탐지기에서 침입 판정을 수행한다. 침입 상황 표시기에서는 탐지 결과를 보여준다.

3.1 로깅 시스템

본 논문에서는 BSM을 이용하여 시스템 호출을 로그 파일에 기록한다. BSM Solaris 2.x 계열의 UNIX 시스템에서 사용하는 시스템 호출 로깅 시스템이다. BSM이 동작되는 전체 구성도는 (그림 2)와 같다. 여기에서 필요한 것은 세 가지로 요약할 수 있다.



(그림 2) BSM에서 audit 처리 관계도

첫째, BSM에서 얻을 수 있는 내용이 무엇인지 분석이 필요하다. 이 내용에 따라 침입 패턴의 분류와 정형화가 이루어진다. audit trail에는 audit 레코드를 기록하는데, 각 레코드는 audit 토큰으로 이루어진다. (그림 3)은 audit trail에서 기록하는 토큰의 종류이다.

• file token • header token • trailer token • arbitrary data token • in_addr token • ip token • iport token • opaque token • path token • process token • return token • subject token	• System V IPC token • text token • attribute token • groups token • System V IPC permission token • argument token • exec_args token • exec_env token • exit token • socket token • sequence token

(그림 3) audit trail의 토큰들

둘째, 로그 내용 중 침입 패턴과 관계가 있는 내용을 선별하는 것이 필요하다. 이것은 첫 번째 내용의 결과에 따라 결정된다. 시스템 호출을 기록은 로그 파일이 커지는 현상이 나타날 수 있기 때문에 침입탐지에 필요한 기록을 분류하는 방법이 필요하다. 또한, 로깅으로 인한 시스템 성능저하를 줄이기 위해서 탐지에 필요한 시스템 호출만을 기록하여야 한다.

셋째, 로그 파일을 침입탐지에 이용하기 쉽도록 가공하는 것과 크기를 줄이기 위해 압축하는 방법이다. 실시간 침입탐지가 이루어져야 하기 때문에 압축하는 방법은 사용하지 않는다. 위의 두 번째 내용에 따라 로그 내용 중 필요 없는 부분을 제거하고 침입탐지에 필요한 자료 구조를 갖도록 하였다.

기존의 침입탐지 시스템에서 시스템 호출을 이용하는 방법은 많았다. 그러나, 대부분 파일의 읽기와 쓰기거나 실행(*exec*)의 경우만 처리하였다. 본 논문에서는 침입의 상황을 정확히 표현하기 위해 필요한 모든 시스템 호출을 이용하였다.

3.2 탐지모듈 생성기

침입탐지에 이용할 모듈을 생성하는 프로그램이다. 이 프로그램은 새로운 패턴이 발견될 때마다 입력하여 새로운 모듈을 생성할 수 있도록 하였다. 또한, 기존의 침입탐지 시스템에서 사용한 RUSSEL[4]와 같은 규칙 생성기와 달리 시스템 호출과 그 조건을 쉽게 기술하도록 하였다.

3.2.1 탐지 모듈 형식

침입 패턴은 탐지 모듈을 생성하기 위한 중요 자료가 된다. 본 연구에서는 침입 패턴을 정형화하여 규칙적인 탐지 모듈이 생성되도록 하고 있다. 침입 패턴 형식을 BNF(Backur-Naur Form)으로 표기한 내용은 (그림 4)와 같다.

일반적으로 규칙기반 침입탐지 방법에 상태전이 다이아그램이 자주 언급된다. 상태전이 다이아그램에서는 트랜지션 분기와 같은 경우를 표현할 수 없으며 이해하기가 복잡하다. Petri-net을 사용한 방법은 여러 가지 상황을 표현하기 쉽고 침입상황을 분석하기에 편리하다.

Petri-net은 여러 가지 구조나 상황을 모델링하고 검증하는 도구로 많이 사용되고 있으며, 침입탐지 분야에서도 침입규칙을 표현하는데 사용하고 있다[1,12,18]. Petri-net은 플레이스(place), 트랜지션(transition), 화살

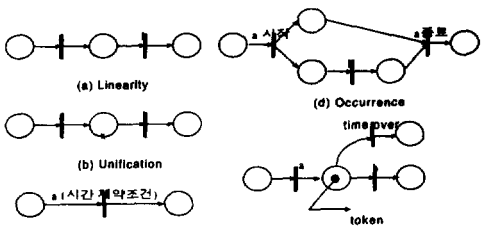
표(arc), 그리고 토큰(token)으로 이루어진다[11]. 본 논문에서는 침입패턴을 Petri-net으로 정형화하여 침입탐지에 이용될 수 있도록 하고 있다. Petri-net의 플레이스(place)는 침입 단계를 의미하고 트랜지션은 침입이 이루어지는 사건을 의미한다. 화살표는 침입의 방향을 표시하고 토큰은 침입의 상황을 표현한다. 현 플레이스에서 해당되는 사건이 발생하면 다음 플레이스로 토큰을 진행시킨다. 침입탐지 시스템에서 사건은 명령어나 시스템 호출과 같은 침입방법에 해당된다. 이와 같은 형식의 입력 패턴은 Petri-net 형태의 침입탐지 모듈로 변경된다. 위 BNF 표기에서 <condition> 부분이 토큰의 진행을 결정하는 부분이다.

```
<pattern_list> ::= /* nothing */
            | <pattern> <pattern_list>
<pattern> ::= <pattern_name> <pattern_body>
<pattern_name> ::= PATTERN ID PSTART
<pattern_body> ::= <event_sequence> PEND
<event_sequence> ::= <event_sequence><event>';'
            | event ';'
<event> ::= <system_call> <condition>
            | EXITON <system_call> <condition>
            | <fork_event> <fork_list> <join_event>
<fork_event> ::= FORK
<fork_list> ::= <fork_list> <case_event>
            <event_sequence> | <event_sequence>
<case_event> ::= CASE
<join_event> ::= JOIN
<system_call> ::= ID '(' INT ')' ':' | INT ':'
<condition> ::= <function> | <condition> OR <function>
            | <condition> AND <function>
<function> ::= <function_name> <parameter>
<function_name> ::= NOT ID '(' | ID '('
<parameter> ::= <parameters> ')' | ')'
<parameters> ::= ID | QSTRING | INT
            | <parameters> ',' ID
            | <parameters> ',' QSTRING
            | <parameters> ',' INT
```

(그림 4) 침입탐지 모듈 생성기 BNF 표기

3.2.2 침입패턴의 특징

침입 패턴은 여러 가지 유형을 가질 수 있다[7]. (그림 5)에서는 그 유형에 따라 어떠한 방법으로 Petri-net을 구성하는지 보여준다. 이것은 탐지 모듈에 대한 Petri-net 형태를 나타낸다.



(그림 5) 침입 패턴의 유형

(a) 선형(Linearity)

사건 발생이 순차적으로 이루어진다. 즉, (그림 5-(a))처럼 사건이 a; b; c 순서로 진행되는 경우이다. 버퍼 오버플로우와 같이 하나의 프로그램으로 침입이 끝나는 경우에 해당된다.

(b) 공통(Unification)

두 사건의 공통 요소가 존재한다. 이전 사건에 사용되었던 매개변수(argument)가 다음 사건에 이용되는 경우이다. 대개 파일을 생성하여 링크시키거나 자료를 편집하여 다른 행위를 취하는 경우가 해당된다. Petri-net으로 구성할 때는 플레이스에서 토큰의 속성에 매개변수를 저장하여 처리한다. (그림 6)에서 ②의 *saveText*()와 ③의 *cmpPath*()가 이에 해당된다.

(c) 시작(Beginning)

사건의 시작 시간에 중점을 두고 있다. 일과시간 이후에 시스템에 접속하는 경우를 예로 들 수 있다. Petri-net에서는 트랜지션에서 사건의 시작 시간을 조사할 수 있도록 처리한다.

(d) 발생(Occurrence)

사건이 진행하는 도중 다른 사건이 발생하는 경우이다. 예를 들어, 어떤 무한루틴 프로그램을 수행시켜놓고 다른 침입 작업을 할 수 있다. 사건은 시작 시간과 종료 시간이 있다. 따라서, (그림 5-(d))에서처럼 사건이 시작해서 종료하기 전까지 다른 사건이 발생할 수 있는 경우를 표현할 수 있다. (그림 6)의 패턴에서 FORK, CASE, JOIN으로 서로 다른 두 개의 프로그램이 동시에 수행되는 것을 표현할 수 있다.

(e) 기간(Duration)

사건의 진행 시간의 범위를 지정할 수 있다. 예를 들어, 어떠한 프로그램은 항상 5분 이내에 작업이 끝나야 한다고 하자. 그런데, 그 프로그램이 5분이 경과하여도 끝나지 않을 경우 문제가 될 수 있다. (그림 5-(e))처럼 time over라는 트랜지션을 두어 일정 시간이 지날 경우 이 트랜지션이 처리되도록 정의한다. 본 논문에서는 감시하는 프로그램의 정상적인 종료를 표현하기 위해서 이 형식을 사용하였다((그림 6)의 ① 참조).

3.2.3 탐지 패턴의 예

탐지패턴의 구조는 아래와 같다.

PATTERN *pattern_name* BEGIN
event_list
END

탐지 패턴이 여러 개 있을 경우 위 구조를 계속해서 나열하면 된다. *pattern_name*에는 패턴에 이름을 기입한다. *event_list*부분은 여러줄 기술될 수 있으며 3가지 형태를 가진다.

첫째, 가장 기본적인 형태는 아래와 같다.

systemcall_number : *conditions* ;

*systemcall_number*는 UNIX 시스템 호출 번호이며, *conditions* 부분은 함수와 관계연산자를 연결하여 표현한다. 예를 들어, 아래와 같이 탐지 패턴을 기술하였다고 하자.

23 : inSTR("/bin/ln") AND NOT isEUID("root");

이 패턴은 시스템호출 23번으로 *execve*(2) 이벤트가 발생하였을 경우에 해당된다. 그리고 처리 조건은 프로그램이 ln이며 effective UID가 *root*가 아닌 경우를 나타낸다.

둘째, 병행수행의 경우가 있는 경우는 아래 구조를 따른다.

FORK
event_list
CASE
event_list
[CASE
event_list
...]
JOIN ;

위에서 CASE는 여러 개 나타날 수 있으며, 그러한 경우 병행 수행하는 프로세스가 여러 개 있게 된다. 위의 *event_list*는 앞서 언급했던 패턴의 *event_list*와 같은 특성을 갖는다.

셋째, 탐지 패턴이 정상적인 것으로 판단되는 경우이다. 사용방법은 아래와 같다.

EXITON *systemcall_number* : *conditions* ;

일반 이벤트 기술방법과 같으나 앞에 "EXITON"이라는 키워드가 들어간다. 실제적으로 사용된 예를 들면 아래와 같다.

EXITON 1 : equalPID();

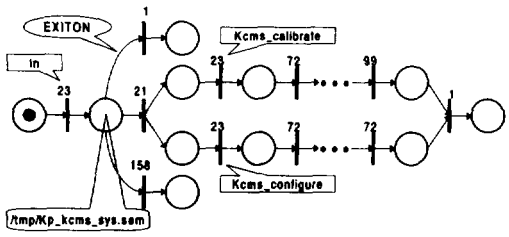
이것은 시스템호출 1번 *exit*(2)가 수행되는 경우로서 해당되는 프로세스가 종료되었을 경우를 뜻한다.

kcms_calibrate를 이용한 임의파일을 생성하려는 공격에 대한 탐지 패턴의 예는 (그림 6)과 같다. 이 공격은 임의 파일을 **/tmp/Kp_kcms_sys.sem**에 연결한 후 **/usr/openwin/bin/kcms_calibrate** 프로그램을 실행한다. **kcms**라는 쉘 스크립트 프로그램은 **kcms_onfigure** 프로그램을 수행하고 임의 파일의 내용을 변경하는 코드가 들어 있다. 이 **kcms**를 경쟁적으로 실행함으로써 **kcms_calibrate**에서 사용하는 임시파일 **/tmp/Kp_kcms_sys.sem** 파일의 퍼미션을 변경하여 임의 파일을 생성하게 된다. 이 공격 방법의 시스템호출을 분석하고, (그림 4)와 같은 형식에 따라 탐지 패턴을 기술하면 (그림 6)과 같이 나타난다.

```
PATTERN ptn_kcms_calibrate BEGIN
    23 : inSTR("/bin/ln") AND NOT isEUID("root");
    EXITON 1 : equalPID();                    ----------①
    EXITON 158 : NOT exist_process();
    21 : inSTR("/tmp/Kp_kcms_sys.sem") AND
        equalPID() AND success() AND saveText(); ---②
    FORK
        23 : inSTR("/openwin/bin/kcms_calibrate")
            AND isEUID("root") AND NOT equalID();
        72 : cmpPath() AND equalPID()
            AND success();
        72 : inSTR("/tmp/logtechpt.sem")
            AND equalPID() AND success();
        108 : equalPID();
        72 : inSTR("/tmp/CP11") AND equalPID()
            AND success();
        104 : equalPID();
        99 : equalPID();
    CASE
        23 : inSTR("/openwin/bin/kcms_configure")
            AND isEUID("root") AND NOT equalID();
        72 : equalPID() AND
    inSTR("/openwin/share/etc/devdata/profiles/kcmsEK");
        72 : equalPID() AND cmpPath() ------------③
            AND success();
        72 : inSTR("/tmp/logtechpt.sem") AND
            equalPID() AND success();
    JOIN;
    1 : equalPID();
END
```

(그림 6) 탐지 패턴의 예

탐지 패턴의 입력은 (그림 6)과 같은 형태로 이루어지며, (그림 7)의 내용은 패턴의 구성을 이해하기 쉽도록 (그림 6)의 탐지 패턴을 Petri-net 형태로 구성해본 것이다.



(그림 7) 탐지 패턴의 Petri-net 구성 예

3.3 침입탐지기

로그 파일과 탐지 모듈을 근거로 하여 침입을 판정하는 중심 프로그램이다. 탐지모듈 생성기에서 입력 패턴을 탐지 모듈로 변환시킨다. 침입탐지기에서는 이 모듈을 사용하여 실시간에 침입탐지를 수행한다.

실시간 처리를 위하여 프로세스 수준에서 검색하고 각 시스템 호출마다 모듈화 하였다. 호스트 기반의 침입탐지 시스템에는 통계적 처리방법과 패턴 인식 방법 등이 주로 이용된다. 본 논문에서는 패턴 인식 방법을 주요 방법으로 하고 퍼지값을 적용하여 다양한 공격에 대응하도록 하였다.

침입탐지기에서는 Petri-net의 플레이스 정보를 저장하고 있다. 이것은 로그 자료를 탐지 모듈에 적용할 때 사용되며 침입 판단에 중요한 역할을 한다. 본 논문에서는 지연공격에 대응하기 위하여 플레이스 상태를 저장하고 있으며, 다중 공격에 대비하여 사용자별 감시를 하지 않고 전체 사용자를 함께 감시하도록 하고 있다. 또한, 알려진 공격 방법에서 조금 변형된 공격에도 대응할 수 있도록 플레이스에 퍼지값을 두었다.

3.4 침입상황 표시기

탐지 결과에 따라 텍스트나 윈도우 상에 표시할 수 있도록 하는 프로그램이다. 침입탐지기에서는 침입 상황을 침입상태 파일에 기록한다. 침입 상황 표시기는 침입상태 파일의 내용을 기반으로 동작한다. 또한, 그 내용에 따라 침입행위를 하는 사용자나 프로그램에 조치를 취할 수 있다.

4. 침입탐지 결정

4.1 변형 공격

변형공격은 일반적으로 알려진 공격방법을 조금 바꾸어 공격하는 형태이다. 일반적으로 침입방법이 한가지 알려지면 그와 비슷한 형태의 공격방법이 발생하기 마련이다. 하나의 공격방법에 대하여 여러 가지 변형이 나타날 수 있어서 하나의 패턴으로 이것을 방어하기는 어렵다. 또한, 시스템 호출의 경우 동일한 프로그램을 실행하더라도 상황과 환경에 따라 시스템 호출이 다르게 나타난다. 일정한 패턴으로 이러한 경우를 수용하는 것은 어렵다.

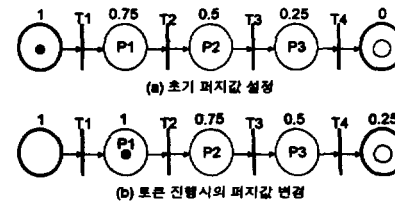
기존의 오용탐지 방법으로는 이러한 경우 탐지할 수 없었다. 여기에 본 논문에서는 이상 탐지의 개념을 접목하여 변형 공격에 유동적으로 대처할 수 있도록 하였다. 그것은 침입 패턴에 대한 Petri-net의 각 플레이스에 퍼지값을 두는 것이다. 이러한 퍼지값은 침입의 진행 정도를 표시하고, 퍼지값에 따라 조금 변형된 공격 패턴도 탐지할 수 있다.

4.2 퍼지값의 적용

먼저 Petri-net의 정의를 살펴보고 퍼지 Petri-net을 정의하여 본다.

Petri-net PN은 다섯 가지 요소를 갖는 $PN=\langle P, T, I, O, M_0\rangle$로 정의된다[11,18]. 여기서 P는 플레이스 집합이며, T는 트랜지션의 집합이다. I와 O는 트랜지션의 입력함수와 출력함수이며 M_0은 초기설정 플레이스이다.

본 논문에서 사용한 퍼지 Petri-net FPN는 $FPN=\langle P, T, I, O, start, final, F\rangle$로 구성하였다. 여기서 F는 각 플레이스가 갖는 퍼지값의 집합이다. 그리고, $start$는 시작 플레이스이고 $final$은 종료 플레이스이다. 위의 FPN의 요소는 각 탐지 패턴마다 저장되며 침입의 진행에 따라 퍼지값이 바뀌게 된다. 플레이스 집합 P는 배열로서 저장되고, 트랜지션 집합 T는 탐지 모듈로 구성하였다. 트랜지션의 입출력 함수 I와 O는 집합 P와 집합 T 사이의 포인터로서 구현하였다.



(그림 8) 퍼지값 설정과 변화

퍼지값을 설정하는 것은 종료 플레이스에서의 상대

적인 거리로 판단한다. 즉, 선형 패턴이 (그림 8)과 같다면, 각 플레이스에서의 초기 퍼지값은 (그림 8-(a))의 플레이스 위의 숫자이다. 퍼지값은 침입이 진행됨에 따라 값이 변경된다((그림 8-(b))). 따라서, 종료 플레이스의 퍼지값도 변경되므로, 이 값에 따라 침입의 진행상황을 판단할 수 있다.

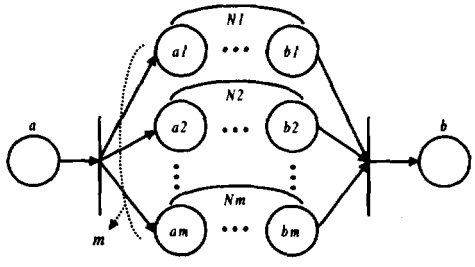
초기 퍼지값은 시작 플레이스를 1.0으로 종료 플레이스를 0으로 설정한다($F_{final}=0$, $F_{start}=1.0$). 이하 플레이스는 시작 플레이스와 종료 플레이스의 거리 비율로 계산된다. 각 플레이스의 퍼지값을 구하는 수식은 아래와 같다(그림 9).

$$F_j = F_a - \frac{F_a - F_b}{N} * n,\quad a < j \leqslant b$$

(그림 9) 선형 패턴

N은 a 플레이스에서 b플레이스까지의 플레이스 거리이고, n은 a 플레이스에서 j플레이스까지의 플레이스 거리이다. F_j 값은 이전 플레이스 F_a과 F_b사이의 플레이스 수와 퍼지값에 따라 결정된다. 트랜지션 분리가 있을 경우 분리에서 결합 사이의 플레이스 수(N)는 달라지는데 그 값은 아래와 같다(그림 10).

$$N = \frac{\sum_{i=1}^{m} N_i}{m},\ m\text{은 분리된 개수이다.}$$



(그림 10) 분리/결합일 때의 퍼지값 설정

퍼지값의 변경은 역시 진행 토큰의 위치와 이웃 플레이스의 증가치에 따라 달라진다. 이벤트가 발생하여 i 플레이스에서 j 플레이스로 토큰이 진행하였을 때, 다음 플레이스의 퍼지값은 선형적으로 증가한다. j 플레이스 이후의 퍼지값(Fk)을 구하는 식은 아래와 같다.

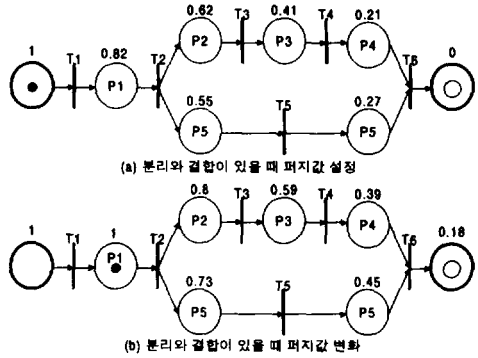
$$F_k = F_k + (1 - F_j),\quad j \leqslant k$$

현재 이벤트가 발생한 j 플레이스의 퍼지값은 1로 되며 다음 플레이스들의 퍼지값(F_k)은 j 플레이스의 증가값만큼 선형적으로 증가한다. 이벤트가 발생한 트랜지션에 결합이 있는 경우 수식은 조금 변경된다.

$$F_k = F_k + \frac{(1 - F_j)}{N_{join}},\quad j < k$$

여기서 N_{join}은 결합된 횟수이다.

변형 공격에 대응하여 각 트랜지션에서는 입력 플레이스에 토큰이 없더라고 그 퍼지값을 비교하여 임계값 이상일 경우 다음 플레이스로 토큰이 진행할 수 있도록 하였다.



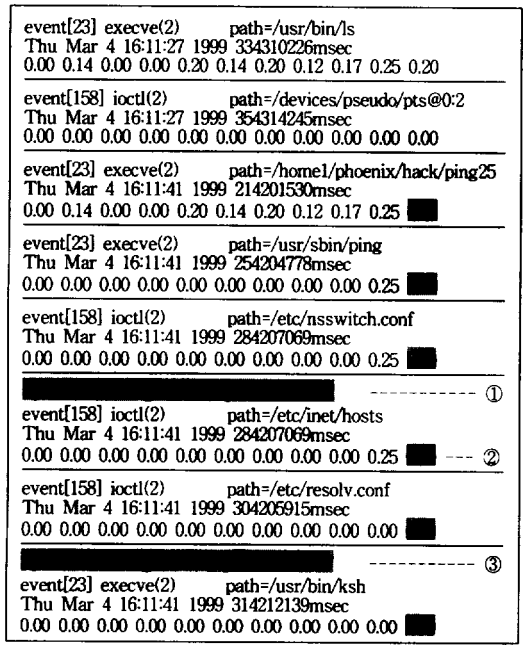
(그림 11) 분리와 결합에서의 변화

4.3 침입 판단

침입탐지기에서 로그 자료를 탐지 모듈에 적용함으로써 침입 여부를 결정한다. 침입의 판단은 종료 플레이스의 퍼지값을 가지고 한다. 퍼지값의 변화는 종료 플레이스의 퍼지값을 계속 상승시키게 된다. 즉, 침입이 진행될수록 종료 플레이스 값은 상승하게 된다. 이 값이 임계치 이상이 되면 침입이라고 판단할 수 있다 ($F_{final} \geq \varepsilon$, $0 < \varepsilon < 1.0$).

침입 판정의 퍼지값을 1.0으로 하지 않고 그보다 작은 임계값으로 한 이유는 침입이 완전히 이루어진 상태($F_{final}=1.0$)에 침입 판정하면 이미 그 순간 다른 작업을 수행하여 시스템에 유해한 상황이 발생할 수 있기 때문이다. 즉, 실시간 탐지를 위해서 퍼지값이 1.0

에 가까워지면 침입이라고 판정하고 대응할 수 있기 위해서이다. (그림 12)에서는 ping 버퍼 오버플로우 공격에 대한 탐지 예제를 보여준다. (그림 12)의 ②처럼 퍼지값이 0.8로서 임계값(여기서는 0.75)을 넘어설 경우 (그림 12)의 ①과 ③처럼 해당 프로세스에게 SIGKILL 신호를 보낸다. 여기에서 신호를 두 번 보내게 된 이유는 공격 프로세스와 CPU경쟁을 하게되기 때문이다.



(그림 12) 버퍼 오버플로우 공격에 대한 탐지 예제

5. 결 론

본 논문에서는 알려진 침입방법에 대한 탐지 규칙을 정형화하여 Petri-net 형태로 구성하였다. 이렇게 생성된 탐지 모듈은 실시간 침입탐지에 사용된다. 탐지 모듈을 생성하는 프로그램은 확장성을 제공하여 새로운 패턴이 발견될 때마다 입력하여 새로운 모듈을 생성할 수 있도록 동작된다. 또한, 다른 운영체제에서도 인터페이스만 정의하면 이용할 수 있다.

본 논문에서는 침입 탐지의 정확성을 높이기 위해 시스템 호출 로깅을 사용하여 로그 자료를 수집하였다. 이것은 탐지 모듈에 적용되어 침입을 판단할 수 있는 근거가 되었다.

변형 공격 등의 다양한 공격 방법에 대응할 수 있도록 퍼지(fuzzy)이론을 적용한 퍼지 Petri-net을 제안하였다. 또한, Petri-net의 플레이스 정보를 저장하여 지연공격과 다중공격에 대응할 수 있도록 하였다.

앞으로 오용공격에 대한 대응방법과 멀티호스트 환경으로 확장할 수 있는 방법에 대한 연구가 필요하다.

참 고 문 헌

[1] M. Crosbie, B. Dole, T. Ellis, I. Krsul, and E. Spafford, "IDIOT - Users Guide," Technical Report TR-96-050, COAST Lab., Sep., 1996.

[2] D. E. Denning, "An Intrusion-Detection Model," IEEE Trans. on Software Engineering, No.2, Feb., 1987.

[3] C. Dowell and P. Ramstedt, "The Computer-Watch Data Reduction Tool," 13th National Computer Security Conference, Oct., 1990.

[4] N. Habra, B. L. Charlier, A. Mounji, and I. Mathieu, "ASAX: Software Architecture and Rule-Based Language for Universal Audit Trail Analysis," Proc. of ESORICS'92, Nov., 1992.

[5] K. Ilgun, "USTAT: A Real-time Intrusion Detection System for UNIX," Proc. IEEE Computer Society Symposium on Research in Security and Provacy, May, 1993.

[6] H. S. Javitz and A. Valdes, "The SRI IDES Statistical Anomaly Detector," Symposium on Research in Security and Privacy, May, 1991.

[7] S. Kumar and E. H. Spafford, "An Application of Pattern Matching in Intrusion Detection," Purdue University, Jun., 1994.

[8] G. E. Liepins and H. S. Vaccaro, "Intrusion Detection: Its Role and Validation," Computers & Security, Nov., 1992.

[9] T. F. Lunt, A. Tamaru, and F. Gilham, "IDES: A Progess Report," Proc. 6th Annual Computer Security Applications Conference, Dec., 1990.

[10] T. F. Lunt, "A Survery of Intrusion Detection Techniques," Computer & Security, Vol.12, No.4, Jun., 1993.

[11] J. L. Peterson, Petri Net Theory and the Modeling of Systems, Prentice Hall, 1981.

[12] P.A. Porras and R.A. Kemmerer, "Penetration state transition analysis: A ruel-based intrusion detection approach," Proc. 8th Annual Computer Security Applications Conference, Nov., 1992.

[13] P. A. Porras, and P. G. Neumann, "EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances," 20th National Information Systems Security Conference, 1997.

[14] P. Proctor, "Audit Reduction and Misuse Detection in Heterogeneous Environments: Framework and Application," Proc. 10th Annual Computer Security Applications Conference, Dec., 1994.

[15] Sandia National Lab., "Intrusion Detection and Response," National Info-Sec Technical Baseline, http://doe-is.llnl.gov/nitb/docs/nitb.html, Oct., 1996.

[16] S. R. Snapp, J. Brentano, G. V. Dias, T. L. Goan, L. T. Heberlein, C. Ho, K. N. Levitt, B. Mukherjee, S. E. Smaha, T. Grance, D. M. Teal, and D. Mansur, "DIDS (Distributed Intrusion Detection System) - Motivation, Architecture, and An Early Prototype," 14th National Computer Security Conference, Oct., 1991.

[17] Sunsoft, *Solaris 2.5 SunShield Basic Security Module Guide*, 1995.

[18] V. Varadharajan, "Petri Net Based Modelling of Information Flow," Proc. The Computer Security Foundations Workshop III, Jun., 1990.

[19] G. B. White and U. Pooch, "Cooperating Security Managers: Distributed Intrusion Detection Systems," Computer & Security, May, 1996.

[20] 노정석, 김휘강, 조용상, 최재철, "특집II - 해킹의 최신 형태와 방지 테크닉", http://www.chosun.com/internetmag/9605/sp2.html, 1996.

[21] 이희조, "인터넷 침입수법/대응방안", NETSEC-KR '97, 1997년 5월.

김 민 수

e-mail : minsoo@chonnam.chonnam.ac.kr

1993년 전남대학교 전산통계학과 (학사)

1995년 전남대학교 대학원 전산통계학과(이학석사)

1995년~현재 전남대학교 시간강사

1996년~현재 전남대학교 대학원 전산통계학과 박사과정

관심분야 : 침입탐지, 시스템보안, 통신보안 등

은 유 진

e-mail : silver@kisa.or.kr

1995년 아주대학교 컴퓨터공학(학사)

1997년 아주대학교 컴퓨터공학(석사)

1996년 12월~현재 한국정보보호센터 기술개발부 연구원

1997년 10월~현재 한국정보통신기술협회 연구위원

관심분야 : 컴퓨터/네트워크 정보보호, 전자서명 인증기술 등

노 봉 남

e-mail : bongnam@chonnam.chonnam.ac.kr

1978년 전남대학교 수학교육학과 (학사)

1982년 한국과학기술원 전산학과 (공학석사)

1994년 전북대학교 대학원 전산통계학과(이학박사)

1983년~현재 전남대학교 컴퓨터정보학부 교수

관심분야 : 통신망관리, 정보보안, 컴퓨터와 정보사회 등