# 침입 시나리오의 페트리 넷 모형

임재걸*
*동국대학교 컴퓨터학과
e-mail:yim@wonhyo.dongguk.ac.kr

# A Petri Net Model for an Intrusion Scenario

Jaegeol Yim
Dept. of Computer Science, Dongguk University at Kyungju

요약

본 논문은 침입 시나리오를 모델하고, 침입의 진행을 시뮬레이션할 수 있는 IDPN을 제안한다. 제안된 IDPN은 기존의 상태변환 다이어그램[1]이나, 퍼지 페트리 넷[2]의 기능을 강화한 것으로, 협동침입과 재침입을 탐지할 수 있고, 침입의 진행에 따른 경고를 발생할 수 있다는 장점이 있다.

## 1. 서론

본 논문은 침입 시나리오를 표현하기 위하여 사용될 수 있는 페트리 넷을 제안한다. 침입 시나리오를 표현하는 방법으로 [1]에는 상태 변환 다이어그램이 소개되었다. 본 논문이 제안하는 페트리 넷은 두 개 이상의 프로세스가 협력하여 침입하는 시나리오를 쉽게 표현할 수 있다는 점에서 상태 변환 다이어그램보다 더 표현 능력이 뛰어나다고 할 수 있다.

탐지된 침입에 대한 처방이 미약하여 침입 상태만 치유하고 침입 진행의 중간 상태를 그냥 방치하는 경우가 종종 있을 수 있다. 그러면 기존의 침입 진행 중간 상태에서부터 재침입이 가능하다. 이런 경우 [2]에서는 재침입 탐지가 불가능하다. 제안된 페트리 넷은 재침입 탐지가 가능하다.

## 2. 관련연구

참고문헌 [1]에는 침입 시나리오를 상태 변환 다이어그램으로 표현하고, 이를 바탕으로 침입을 탐지하는 방법이 소개되었다. 본 논문은 상태 변환 다이어그램으로 표현할 수 있는 침입 시나리오를 모델링할 수 있는 페트리 넷을 제안하는 것이 목적임으로 본 절에서는 [1]의 내용을 소개한다.

NCSC(미국 컴퓨터 보안 센터)가 C2 급 이상으로 평가하는 운영체제는 감사(audit) 수집 기능을 반드시 제공한다. UNIX 운영체제에는 BSM(기본 보안 모듈)이라는 모듈이 있어, 감사 수집을 포함한 보안 기능을 수행한다. 감사 수집이란 시스템에서 수행되는 모든 동작(action)을 다음과 같은 형식의 레코드로 기록하는 것이다:

<주체, 동작, 객체>

이 레코드 형식은 다음과 같은 의미를 표현한다: "'주체'가 '객체'에 '동작'을 수행하였다." '주체'는 다음과 같은 형식으로 구성된다:

<사용자 ID, Effective 사용자 ID, Group ID>.

동작은 다음과 같은 형식으로 표현된다:

<동작, 시간, 프로세스 ID>.

객체는 다음과 같은 형식으로 표현된다:

<객체 이름, 허용, 주인, 그룹 주인, Inode #, 장치 #, 파일 시스템 ID, Target>.

객체 이름은 파일의 경로이고, Target은 동작이 hardlink나 rename일 경우에만 쓰인다.

BSM에 의하여 기록되는 감사 레코드에는 239 가지 동작들이 있는데, 그 중 28가지가 침입에 관련된 것이며, 이들은 10 가지 유형으로 분류된다. 즉, 상태 변환 다이어그램에 나타나는 동작은 이들 10 가지뿐이다. 이 유형은 <표 1>과 같다[3].

| 동작유형 | BSM 동작 유형 |
|---|---|
| Read | open_r, open_rc, open_rtc, open_rwc, open_rwtc, open_rt, open_rw, open_rwt |
| Write | truncate, ftruncate, creat, open_rwc, open_rwtc, open_rw, open_rwt, open_rt, open_rtc, open_w, open_wt, open_wc, open_wtc |
| Create | mkdir, creat, open_rc, open_rtc, open_rwc, open_rwtc, open_wc, open_wtc, mknod |
| Delete | rmdir, unlink |
| Execute | exec, execve |
| Exit | exit |
| Modify_Owner | chown, fchown |
| Modify_Perm | chmod, fchmod |
| Rename | rename |
| Hardlink | link |

<표 1> 동작의 유형 분류 표

[1]에서는 파일들도 침입과 관련된 성질에 따라 <표 2>와 같이 분류한다.

[1]에서는 상태를 표시하기 위하여 다음과 같은 11 가지 부울 함수를 사용한다. 각 함수의 의미는 함수명에서 유추할 수 있으며, 자세한 사항은 [1]을 참고하기 바란다. 이들 함수를 논리연산자로 연결한 식으로 상태를 단언한다. 본 논문에서는 이러한 식을 '조건식'이라고 한다. (*Petri net* 정의에서 '조건식'이라는 용어를 사용함.)

name(file_var)=file_name, fullname(file_var)=full_path,

| 파일 유형 | 성질 |
|---|---|
| Fileset #1 | 특정인만 읽을 수 있는 파일들 |
| Fileset #2 | 특정인만 쓸수 있는 파일들 |
| Fileset #3 | Fileset #1에 대한 읽기 권한이 있는 파일들 |
| Fileset #4 | Fileset #2에 대한 쓰기 권한이 있는 파일들 |
| Fileset #5 | 쓰기 금지, 시스템 수행 파일들 |
| NWSD | 쓰기 금지, 시스템 디렉토리 |
| HARDLINK | 시스템 hardlink 정보 |

<표 2> 파일 유형

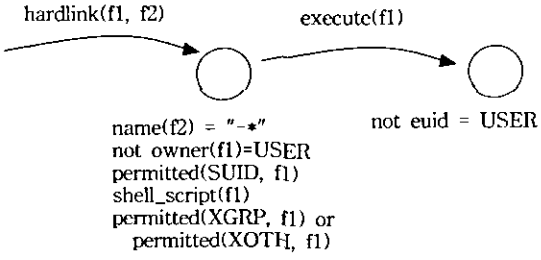
owner(file_var)=user_id, member(file_set, file_var), euid=user_id, gid=group_id, permitted(perm, file_var), located(NWSD, file_var), same_user, same_pid, shell-script(file_var).

예를 들어, euid=user_id는 처리 중인 감사 레코드의 주체의 '사용자 id'가 user_id와 같으면 참이다.

상태변환 다이어그램의 예로 <그림 1>을 들 수 있다. 이 그림은 다음과 같은 단계를 거쳐 침입하는 것을 나타내는 다이어그램이다.

%ln f1 f2
%f2

단, f2는 '-'로 시작하는 파일 이름이고, f1은 root의 setuid 쉘 스크립트 파일로서 첫줄이 '#!/bin/sh'와 같이 쉘을 수행하는 명령이며, 그룹이나 기타 사용자가 수행할 수 있도록 허용된 파일일 때 이 침입 시나리오는 성공적으로 수행될 수 있다.



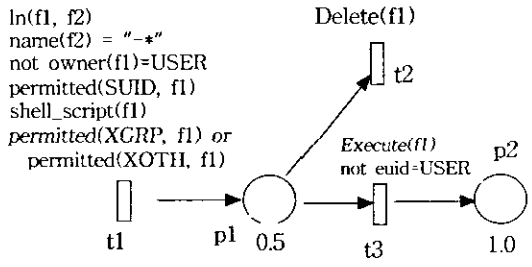

<그림 1> 상태 변환 다이어그램의 예

### 3. 제안된 페트리 넷

제안하는 페트리 넷은 상태 변환 다이어그램으로 표현할 수 있는 모든 침입 시나리오를 나타낼 수 있을 뿐 아니라, 실제 침입의 진행 상황을 페트리 넷 상에 나타낼 수 있음으로써 침입 탐지에 사용될 수 있어야 한다. 그렇게 하면, 페트리 넷 특성상 둘 이상의 침입자나, 둘 이상의 프로세스가 서로 협력하는 침입 시나리오도 표현이 가능하게 된다.

제안하는 페트리 넷의 정의를 소개하기 전에 <그림 1>과 같은 상태 변환 다이어그램을 제안하는 페트리 넷으로 표현하여 보면 <그림 2>와 같다.

<그림 2>를 바탕으로 제안하는 페트리 넷을 설명하면 다음과 같다. 페트리 넷은 그림과 같이 사각형과 원들, 그리고 이들을 잇는 간선으로 구성된다. 사각형을 transition이라고 부르고, 각 transition은 t1, t2, t3과 같은 고유 이름을 갖고 있다. 원은 place라고 하며, 각 place는 p1, p2와 같은 고유 이름을 갖는다[4]. Transition과 place들을 모두 합하여 정점 V라 하고, V의 원소를 v라 하면, $(v_1, v_2)$가 간선일 때, $v_1$은 $v_2$의 입력 transition(혹은 place)라 하고, $v_2$는 $v_1$의 출력 place(혹은 transition)이라 한다.




<그림 2> 그림 1의 페트리 넷 모형

본 페트리 넷에서 transition은 사건을 나타내며, place는 사건의 결과 처하게 되는 상태를 나타낸다. 즉, <그림 2>에서 p1은 앞에서 언급한 조건을 만족하는 'ln f1 f2'라는 명령이 성공적으로 수행된 상태를 의미하며, p2는 p1 상태에서 root가 아닌 사용자가 root의 권한으로 f1을 수행하는 침입 상태에 있음을 나타낸다.

p2는 침입 상태를 나타냄으로 침입을 100% 단언한다는 의미에서 1.0이라는 레벨을 갖는다. 또한 p1은 침입 시나리오가 두 단계로 구성되었는데 그 중 처음 단계가 수행된 상태임으로 침입이 50% 진행되었다는 의미에서 0.5라는 레벨을 갖는다.

침입의 현재 진행 상태를 나타내기 위하여 토큰을 해당 place에 놓는다. 일반적으로 토큰은 까만 점이다. 그러나, 본 페트리 넷의 *토큰은 감사 레코드의 주체와 객체*에 대한 정보로 구성된다. <그림 2>에는 현재 아무런 토큰도 없다. 즉, 이 그림이 나타내는 침입 시나리오가 전혀 진행된 바 없음을 나타낸다.

어떤 place에 토큰을 놓고, 어떤 place의 토큰을 제거하는가는 transition의 *격발 규칙*에 의하여 결정된다. 본 페트리 넷의 경우 transition $t_i$의 격발 규칙은 $t_i$의 출력 place가 있는가 없는가에 따라 다르다.

1) 출력 place가 있는 경우: 모든 입력 place에 토큰이 있고, 현재 처리 중인 감사 레코드가 $t_i$에 쓰인 동작 및 '조건식'을 만족할 때 격발한다. $t_i$의 격발은 입력 place의 토큰을 그대로 두고 출력 place에 새로운 토큰을 첨가한다.

2) 한편, $t_i$의 출력 place가 없는 경우에는 격발 결과 입력 place의 $t_i$의 조건식을 만족하기 위하여 사용된 토큰을 제거한다.

*격발 예*: <그림 2>에서 격발의 예를 보이면 다음과 같다. 감사 레코드가 $t_1$에 쓰인 동작과 '조건식'을 만족하면, 즉 동작이 'ln f1, f2'이고 f2의 이름이 '-'로 시작하며, ..., 등이 모두 참이면 $t_1$은 격발한다. 왜냐하면 $t_1$의 입력 place가 없음으로 '모든 입력 place에 토큰이 있다'는 말은 공허한 참이기 때문이다. $t_1$의 출력 place는 $p_1$임으로 $p_1$에 토큰을 놓음으로써 현재 상태가 침입 시나리오의 첫 단계가 진행된 상태임을 나타낸다. 토큰은 감사 레코드의 주체와 객체로 구성된다. 즉, p1에 놓이는 토큰은 다음과 같다: <사용자 ID, ..., f1, f2, ...>. 이때 만약 다른 사람이 'ln f3, f4'를 수행하고 이것도 $t_1$의 '조건식'을 만족하면 이 감사 레코드에 대한 주체와 객체로 구성된 토큰, <사용자 ID, ..., f3, f4, ...>가 p1에 가미되어 결과적으로 p1에는 두 개의 토큰이 놓인다.

이 상태에서 'Delete f3'이 수행되면 $t_2$가 격발하고, 격발 결과 <사용자 ID, ..., f3, f4, ...>는 삭제된다. 즉, 'ln f3, f4'로 시작되는 침입 시나리오의 위험이 사라지게 되는 것이다. 그래도, p1에는 'ln f1, f2'를 수행한 결과 생긴 토큰, <사용자 ID, ..., f1, f2, ...>은 아직 있다. 이 상태에서 'Execute f1'을 root가 아닌 사용자가 수행하면 토큰 <사용자 ID, ..., f1, f2, ...>를 사용하여 transition $t_3$이 격발한다. 이때에는 $t_3$의 출력 장소가 있음으로 사용된 토큰을 삭제하지 않고 그대로 둔 채 새로운 토큰을 $p_2$에 하나 더 놓는다. 즉, $p_1$에 놓인 토큰은 $t_2$를 격발하지 않는 한 제거되지 않는다.

따라서, 침입이 탐지된 상태에서 만일 f1을 수행하는 process를 kill하는 조치를 취하더라도, p1에 있는 토큰은 제거되지 않았음으로 추후에 f1을 또 수행하면 다시 침입 상태로 된다.

제안된 페트리 넷을 침입탐지 페트리 넷(IDPT)이라고 칭하면, IDPT의 정의는 <표 3>과 같다.

| |
|---|
| IDPT = (P, T, E, 주체, 동작, 객체, FT, FP)<br>P, T, E는 일반적인 페트리 넷이고, 주체, 동작, 객체는 감사 레코드에 기록된 정보임.<br>FT는 transition에 '동작유형'과 '조건식'을 사상하는 함수 (주체, 동작, 객체를 상태 변환 다이어그램처럼 동작유형과 조건식으로 변환함)<br>FP는 침입 시나리오의 전체 길이(L)를 1로 하여 각 place에 침입 진행 거리($d_i$)의 비율($d_i$/L)을 표시하는 함수. 경로가 여럿(n 개)일 경우에는, 이 비율을 n으로 나눈 결과를 표시함.<br><br>격발규칙: 본문 참조. (특히 토큰이 주체와 객체로 구성됨에 유의) |

<표 3> IDPN의 정의

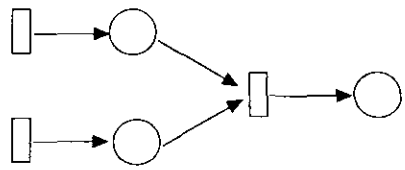
## 4. 기존 연구와 비교 및 사용 방법

제안된 페트리 넷은 참고문헌 [1]에 소개된 상태 변환 다이어그램보다 표현 능력이 작지 않다. 즉, 상태변환 다이어그램의 arc와 원을 각각 IDPN의 transition과 place로 사상하고, 조건식과 동작을 모두 <그림 2>와 같이 *transition*에 사상함으로써, 모든 상태 변환 다이어그램은 IDPN으로 표현될 수 있다. IDPN의 장점으로 다음을 들 수 있다.

1) 상태변환 다이어그램에서는 감사 기록을 분석하여 침입을 탐지하는 작업이 직접 수행되지 않고, 표를 따로 만들어 침입 시나리오의 진행을 분석하여야 한다. 이에 반하여 IDPN에서는 침입 시나리오의 진행을 직접 시뮬레이션하여 탐지한다는 장점이 있다. 이를 위하여 복잡한 격발 규칙과 복잡하게 구성된 토큰을 정의함.

2) IDPN의 각 place에는 침입의 진행 정도가 기록되어 있음으로 어느 정도 이상 침입이 진행되었을 때 경보를 보내는 기능을 쉽게 구현할 수 있다는 장점이 있다.

3) 두 명 이상, 혹은 둘 이상의 프로세스가 협동하여 침입하는 시나리오를 상태변환 다이이그램으로는 표현이 불가능하지만 IDPN으로는 표현이 가능하다. 협동 시나리오의 IDPN 표현은 <그림 3>과 같은 골격으로 표현된다.



<그림 3> 협동 침입 모형의 골격

IDPN의 토큰은 <주체, 객체>로 구성되어 있고, 격발에서 토큰을 소모하지 않을 수 있다는 점이 기존의 연구 [2]와 크게 다르다. 이러한 점 때문에 각 침입 시나리오에 대하여 단 하나의 IDPN만 구축하여 놓으면 여러 사람에 의하여 그 침입 시나리오가 여러 번 수행되어도 모두 탐지할 수 있다. 또한, 탐지된 침입에 대한 처방이 미약하여 침입 상태만 치유하였을 경우, 기존에 진행된 상태에서 재침입을 시도할 수 있는데, 이런 경우 [2]에서는 탐지가 불가능하지만 IDPN에서는 가능하다는 장점이 있다.

향후에는 IDPN을 이용한 침입 탐지 시스템을 구축하고 성능을 분석하는 실험을 수행하고자 한다. 이를 위하여 알려진 모든 침입 시나리오를 IDPN으로 표현한다. 앞에서도 언급한 바와 같이 하나의 침입 시나리오에 하나의 IDPN을 구축하면 족하다.

다음은 감사 기록을 읽고 침입 패턴이 발생하면 IDPN에서 침입 진행을 시뮬레이션하는 모듈을 구현한다. 시뮬레이션 모듈은 IDPN의 모든 enable된 transition을 격발 가능한가 검사한다. Enable된 transition이란 모든 입력 place에 토큰이 들어있는 transition을 일컫는다. 격발 가능한 transition이란 처리 중인 감사 기록이 사상된 조건식을 만족하는 enable된 transition을 일컫는다. 격발 가능한 transition을 찾을 때, enable된 transition 중 침입 상태와 가까운 transition 순서로 검사함으로써 침입 상태를 재빨리 탐지하도록 한다. 시뮬레이션 모듈은 격발 가능한 transition을 격발 규칙에 따라 격발하여 침입 시나리오의 진행을 IDPN에 표시한다.

## 5. 결론

본 논문은 침입 시나리오를 모델하고, 침입의 진행을 시뮬레이션할 수 있는 IDPN을 제안하였다. 제안된 IDPN은 기존의 상태변환 다이어그램[1] 보다 표현 능력이 뛰어나다. 즉, 모든 상태변환 다이어그램은 IDPN으로 표현될 수 있고, 더 나아가서 협동침입 시나리오까지도 IDPN으로 표현할 수 있다. 또한 기존의 퍼지 페트리 넷[2]에서 간과한 재침입도 IDPN에서는 탐지할 수 있다는 장점이 있다.

## 6. 참고 문헌

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

[2] 김민수, 은유진, 노봉남, "UNIX 환경에서 퍼지 Petri-net을 이용한 호스트 기반 침입 탐지 모듈 설계," 정보처리논문지, 제6권 제7호, pp. 1867-1876, 1999.

[3] K. Ilgun, "USTAT: A Real-time Intrusion Detection System for UNIX," Master's Thesis, Computer Science Dept., University of California, Santa Barbara, Nov. 1992.

[4] Tadao Murata, "Petri Nets: Properties, Analysis and Application," Proceedings of the IEEE, Vol. 77, No. 4, April 1989, pp. 541-580.