성능 시그네쳐를 이용한 서비스 거부 공격 침입탐지 시스템 설계

김 광 득† · 이 상 호††

요 약

서비스 거부 공격은 인가를 얻지 않고 전체 시스템을 붕괴시키는 방법으로 정상적인 시스템 서비스를 방해하는 것이다. 이러한 유형의 공격은 손쉽게 이루어질 수 있으며, 방어하기가 힘들다. 기본적인 문제는 Unix는 자신이나 다른 시스템상의 사용자들이 정상적인 행위를 할 것이라는 가정에 기인한다. 이 논문에서는 내부 서비스 거부 공격과 각 시스템에서 제공하는 성능 메트릭을 분석하여 관측에 따른 성능 영향을 조사하고 그 결과를 시스템 및 프로그램 행위에 대한 시그네쳐로 형식화하는 방법으로 서비스 거부 공격을 탐지하는 새로운 접근법을 제시한다. 이 메트릭은 공격을 탐지하기 위한 자동화된 프로그램 개발 가이드가 될 것이며, 이 결과를 기반으로 성능 시그네쳐를 이용한 서비스 거부 공격을 탐지하는 AIDPS 모델을 제안한다.

Intrusion Detection System for Denial of Service Attack using Performance Signature

Kwang-Deuk Kim† · Sang-Ho Lee††

ABSTRACT

Denial of service is about knocking off services, without permission for example through crashing the whole system. This kind of attacks are easy to launch and it is hard to protect a system against them. The basic problem is that Unix assumes that users on the system or on other systems will be well behaved. this paper analyses system-based inside denial of services attack(DoS) and system metric for performance of each machine provided. And formalize the conclusions results in ways that clearly expose the performance impact of those observations. So, We present new approach. It is detecting DoS attack using performance signature for system and program behavior. we believe that metric will be to guide the automated development of a program to detect the attack. As a results, we propose the AIDPS(Architecture for Intrusion Detection using Performance Signature) model to detect DoS attack using performance signature.

1. 서 론

오늘날 인터넷 기술의 급격한 성장과, 인터넷을 통한 조직과 개인의 사회적 활동의 증가에 따라 인터넷에 대한 우리들의 생활의 의존도가 점차로 증가하고 있다. 이 같은 관점에서 볼 때, 인터넷상의 새로운 위협으로 등장한 서비스 거부(denial of service) 공격, 즉 시스템의 자원을 고갈시켜 정상적인 동작을 방해하여 사용자에 대한 서비스의 제공을 불가능하게 만드는 공격 또한 기존의 다른 어떤 종류의 위협에 못지 않게 심각한 것이라고 할 수 있다. 그 한 예로 1996년 9월 인터넷상의 수십 사이트들이 SYN 플러딩이라 불리는 서비스 거부 공격을 당했다. 이 공격은 TCP/IP(Transmission Control Protocol/Internet Protocol) 프로토콜의 약점을

† 준 회 원 : 한국에너지기술연구소 선임연구원
†† 종신회원 : 충북대학교 컴퓨터과학과 교수
논문접수 : 1999년 6월 28일, 심사완료 : 1999년 9월 27일

이용하는데, 이 공격은 프로토콜의 명확한 수정 없이는 해결될 수 없다. 서비스 거부 공격은 시스템의 주요 파일을 훼손시켜 시스템의 동작을 방해하는 등의 간접적인 서비스거부공격 방법, 시스템 내부 자원 또는 네트워크 관련 자원을 소모시켜 서비스를 제공하지 못하게 하는 직접적인 서비스 거부 공격수법으로 나뉘어 질 수 있으며, 계속하여 새로운 수법들도 발표되고 있다. 침입탐지는 이러한 컴퓨터의 자원의 무결성, 기밀성 그리고 가용성등을 저해하는 행위를 탐지하기 위한 기술을 말하며[6] 탐지 능력을 향상시키기 위해 다수의 에이전트가 공동 대응할 수 있는 탐지기술에 많은 연구가 진행되고 있다[4,7]. 이 논문에서는 직접적인 서비스 거부 공격에 대응할 수 있는 시스템을 구현하기 위해 각 시스템에 대한 메트릭 아이템들을 조사·분석하여 실세계 환경에서 검증하였으며, 이를 이용하여 침입을 탐지 할 수 있는 AIDPS(Architecture for Intrusion Detection using Performance Signature)모델을 제시하였다. 논문의 구성은 2장에서는 직접적인 서비스 거부 공격을 위해 목표로 삼고 있는 시스템의 요소가 어떤 것인 지와 메트릭을 이용한 측정 결과를 분석하였으며, 3장에서는 성능 시그네쳐를 생성하는 방법과 이용하여 이들 공격을 감지할 수 있는 지를 분석하고, 4장에서는 라우터와 호스트 모니터 그리고 네트워크 모니터와 상호 협력을 통하여 침입에 반응 할 수 있는 침입탐지 시스템을 설계하고 그 모델을 제안한다. 마지막장에서는 향후 연구방향과 결론을 제시한다.

2. 공격 목표 및 메트릭 분석

2.1 공격 목표

서비스 거부 공격은 시스템의 자원을 고갈시켜 정상적인 서비스를 방해하는 공격으로 내부공격과 외부 공격으로 대별될 수 있다. 본 장에서는 이러한 공격을 위해 목표가 되는 시스템 자원은 스왑공간, 대역폭, 램, 디스크, 커널 테이블, 캐쉬등이 있으며 이들 중 대표적인 것만 살펴본다.

2.1.1 스왑 공간

시스템은 모든 프로세스를 처리하기 위해 충분한 메모리를 갖지 못할 때, 프로세스 데이타(메모리 메이지)를 디스크로 교체한다. 스왑핑 활동은 스왑(Swap) 영역의 데이터를 메모리로 또 그 반대로 이동시키는 시간 때문에 시스템의 성능을 저하시킨다. 대다수 시스템들은 클라이언트 요구를 서비스하기 위해 수백 메가바이트 스왑 공간을 두는데, 보통은 불러 드려진 자식 프로세스를 처리하기 위해 짧은 시간동안만 이용된다. 그래서 스왑 공간은 일반적인 과중한 부하 문제를 거의 야기하지 않는 것으로 인식하기 때문에 공격자는 이점에 주목한다. 공격자는 지속적으로 프로세스에 대한 스왑 공간을 할당하도록 많은 프로세스를 동시에 반복적으로 수행시킴으로서 스왑 공간을 고갈시킨다. 스왑공간 오버플로우 현상이 나타나면 시스템은 스왑 공간을 확보하기 위해 새로운 프로세스를 제거하기 위한 프로세스를 구동시키게되고, 정상적인 프로세스 처리가 더 이상 진행되지 못함으로서 서비스 거부 현상이 발생된다.

2.1.2 커널 테이블

커널 테이블(kernel table)은 동시에 수행될 수 프로세스의 수를 통제하며 캐쉬와 작은 쓰기 버퍼를 통해 쓰기를 하는 시스템은 커널 메모리 할당하게 된다. 커널은 커널 맵에 한계를 가지며, 만일 이 한계에 도달하면 프로세스가 완전히 끝날 때까지는 다른 프로세스에 대한 메모리를 더 이상 커널 할당할 수 없으며 시스템은 재부팅 되어야 한다. 커널 메모리는 RAM, CPU, 스크린 등에 이용될 뿐만 아니라, 보통의 프로세스에도 이용된다. 솔라리스 2.X 경우에는 얼마나 많은 커널 메모리가 시스템에 사용되는지를 sar 명령어로 측정하여 보고되었지만, SunOS 4.x에서는 그러한 명령어가 없다. 이는 SunOS 4.X하에서는 그러한 데이터를 얻을 수 없음을 의미한다. 만일 솔라리스를 이용한다면 정보를 얻기 위해서는 sar -k 1을 사용하면 된다. 또한 netstate -k가 이용될 수 있는데, 이는 subpaging에 할당하는 커널에 메모리가 얼마인지를 알 수 있다. 이렇게 획득한 정보를 이용하여 공격에 활용한다.

2.1.3 캐쉬

캐쉬(Cache)를 수반하는 서비스 거부 공격은 캐쉬를 차단하거나 또는 제거하는 방법을 이용하여 정상적인 서비스를 방해하는 공격을 가할 수 있는데, 솔라리스 시스템의 경우 공격 대상이 되는 캐쉬는 다음과 같다.

- *Directory name lookup cache* : vnode와 관련한 file의 이름.
- *Inode cache* : 재 참조가 필요로 할 경우 디스크에서 캐쉬 정보를 읽음
- *Rnode cache* : NFS filesystem 에 관한 정보를 유지.
- *Buffer cache* : disk I/O와 관련 블록과 실린더의 간접적인 inode 캐쉬

여기서 Directory name lookup caches는 보통 프로세스 크기에 100을 더한 크기를 가지며, 큰 이름 캐쉬는 룩업을 회피하게 되어 시스템의 과부하를 야기 시키게 된다.

2.2 시스템 메트릭 분석

2.2.1 시스템 메트릭 조사

이 논문에서 제시하려는 성능 시그네쳐를 이용한 침입탐지 기법은 시스템과 프로그램의 정상적인 성능을 조사하여 이를 탐지에 이용하는 방법으로 프로그램과 시스템의 서비스 거부 공격에 대한 예외적인 행위 탐지는 우선 관찰되는 변수의 정의가 요구된다. 또한 관련되는 변수들의 그룹과 각각의 변수에 대해 의심스러운 행위의 기준이 정의되어야 한다. 이는 불법을 표현하기 위해 고려되는 값 또는 이 값의 정상 패턴에서 의미변경을 탐지하는 통계적 모델의 범위를 형성하게 된다. 성능 모니토링과 관계하는 통계는 예외적인 시스템 성능 시그네쳐를 탐지할 수 있는 유용한 메트릭을 구성하기 위하여 <표 1>과 같이 각 시스템에서 제공되고 있는 성능 측정 메트릭들을 분석하였다. 정확하게 말하면 운영 시스템 및 프로그램은 확실한 범위 내에서 이러한 자원을 이용하는데, 이를 이용하여 성능 시그네쳐를 생성하고 침입탐지 시스템에 활용될 수 있는지를 분석하기 위해 알파 서버 4100과 썬 울트라

〈표 1〉 시스템 메트릭 항목

메트릭 이름	메트릭 유형	플 렛 폼						설 명
		Solaris 2.x	Digital UNIX 3.2	HP-UX 9.04 10.x	NCR (AT&T)	AIX 4.1	U6000 1.2	
Pid	정수	•	•	•	•	•	•	프로세스 번호
PPid	"	•	•	•	•	•	•	부모 프로세스 번호
User	문자열	•	•	•	•	•	•	실 사용자 이름
TTY	"	•	•	•	•	•	•	테미널 번호
Group	"	•	•	N/A	•	•	•	실 그룹 이름
Command	"	•	•	•	•	•	•	프로세스 이름
Arguments	"	•	•	•	•	N/A	•	커멘드가 가지고 있는 전체 커멘드 라인
% Usr	실수	•	•	•	•	•	•	사용자가 이용한 CPU 시간 백분율
% Sys	"	•	•	N/A	•	•	•	시스템이 이용한 CPU 시간 백분율
CUsr	부호없는 정수	•	•	N/A	•	•	•	자식 사용자가 이용한 CPU 시간
CSys	"	•	•	•	•	•	•	자식 시스템이 이용한 CPU 시간
RssK	"	•	•	•	•	•	•	이용된 메모리(resident set size)
VMemK	"	•	•	•	•	•	•	전체 가상 메모리 양
Pri	정수	•	•	•	•	•	•	현 타스크 우선순위
Nice	"	•	•	•	•	•	•	프로세스 nice 값
BRead	부호없는 정수	•	•	•	•	•	•	버퍼를 읽은 횟수
BWrit	"	•	•	•	•	•	•	버퍼에 쓰기를 한 횟수
CPU#	정수	•	•	•	•	•	•	현 CPU 프로세서 번호
KChar	부호없는 정수	•	N/A	•	•	N/A	•	전송된 글자의 수
ICSW	"	•	•	•	N/A	•	N/A	무의식중에 스위치된 문맥 수
VCSW	"	•	•	•	N/A	•	N/A	의식적으로 스위치된 문맥 수
PgFlt	"	•	•	•	N/A	•	N/A	페이지 폴트의 수
ShMem	"	N/A	•	•	N/A	•	N/A	공유된 메모리의 이용율
NThreads	"	•	•	N/A	N/A	•	N/A	스레드의 바호
% Cpu	실수	•	•	•	•	•	•	이용된 전체 CPU 시간 백분율
% Mem	"	•	N/A	N/A	N/A	•	N/A	실제 이용된 메모리 백분율

II 시스템을 이용하였다.

2.2.2 성능 시그네쳐 수집

<표 1>에서 나타난 메트릭을 이용하여 측정한 데이타를 침입탐지 시스템의 성능 시그네쳐로 활용될 수 있는지를 분석하기 위하여 필요한 메트릭을 이용한 이벤트를 정의한다. 이 이벤트는 단일 메트릭 변수 또는 연산자를 이용한 변수들의 수식 표현법일 수도 있으며, 메트릭을 이용한 이벤트 표현 문맥, 그리고 관련된 용어는 다음과 같다.

```
(Syntax)
  {Event
   ATTRIB Id
     {CLASS Name
     COLUMN "C_name" = "Process Info.Processes
      :
      :
     end CLASS}
      :
      :
   end Event}
```

(용어)
- 연산자 : AND, OR, *, /, +, - 등
- 표현식 : 연산자를 이용하여 두 개 이상의 변수들의 결합
- 선 택 : 표현식의 그룹
- Column : 프로세스 정보 테이블에 있는 메트릭리스트(표 1 참조)
- 영 : 선택식에서 정의된 기준을 통과한 결과 메트릭
- 테이블 : 행과 열의 그룹인 메트릭 배열

이벤트 범위를 정의할 때 데이타 수집 및 분석을 용이하게 하기 위하여 하나의 이벤트에 여러 개의 클래스를 둘 수 있도록 하였다. 즉 공격에 따른 특정 이벤트가 I/O 한계 속성을 지닌다면, I/O 한계를 조사하기 위한 클래스와 이를 확인하기 위한 클래스로 나누어 정의할 수 있게 함으로서 환경변화에 적절히 대응할 수 있도록 하였다. 이에 대한 예는 다음과 같다.

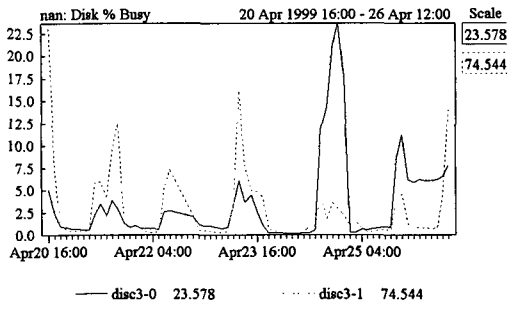
```
(예)
  EVENT
    ATTRIB IO_Event
      CLASS IO_Item
        COULMN " Pid" = "Process Info.Processes.Pid"
        COLUMN "User" = "Process Info.Processes.User"
                :
                :
      COLUMN "Bread" = "Process Info.Processes.Bread"
      COLUMN "Bwrite" = "Process Info.Processes.Bwrite"
    end CLASS
    CLASS Identification_Item
      COLUMN "User" = "Process Info.Processes.User"
                :
                :
      COLUMN "Command" =
                "Process Info.Processes.Command"
      COLUMN "Group" = "Process Info.Processes.Group"
      COLUMN "Ppid" = "Process Info.Processes.Ppid"
      COLUMN "TTY" = "Process Info.Processes.TTY"
    end CLASS
  end EVENT
```

2.3 측정 결과 분석

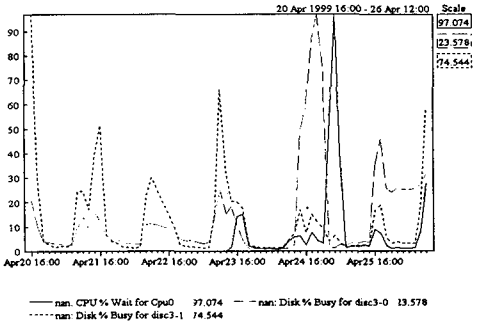
앞절에서 설명한 이벤트들을 정의하고, 실 시스템에 적용하여 측정하여 수집된 자료가 성능 시그네쳐로 활용될 수 있는지를 분석한다.

2.3.1디스크 이벤트

디스크 활용을 측정하기 위한 한 예로 "DISK %Busy" 이벤트를 이용한다. 이것은 I/O 전달이 일어나는 수와 디바이스의 탐색 시간에 의존하는 것으로 이는 최초의 디바이스의 정상 값을 가지고 이상을 판단한다. 측정값이 정상 수치를 넘어서면 I/O 성능에 피해를 주게된다. (그림 1)은 디스크 디바이스 disc3-0과 disc-1을 측정한



(그림 1) DISK Busy 상태



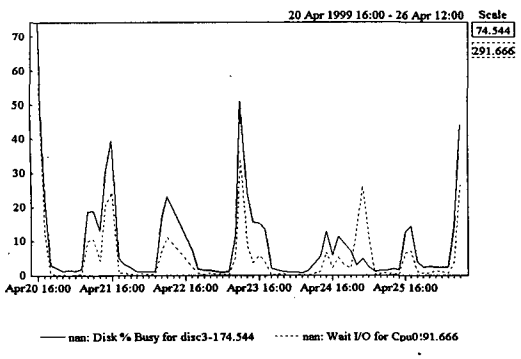
(그림 2) Disk Busy와 타 요소간 연관성 분석

결과를 도시한 것으로 4월24일 14시 30분 경에 디스크 활동이 분주한 상태에 이런 것을 알 수 있다.

그리고 이는 "Disk I/O wait/Sec"과 "Disk Avg Serv"에 직접적인 영향을 주며. 이러한 상황이 지속적으로 발생되면 I/O 바운드를 야기 시킨다. 다음은 이를 입증하기 위한 측정 결과를 (그림 2)에 도시하였다.

2.3.2 CPU 이벤트

CPU에 대한 시스템의 행위를 분석하기 위해 CPU % Wait를 분석한다. 이는 봉쇄된 I/O에 대해 CPU가 대기하는데 소비되는 시간의 백분율이다. 이는 네트워크 파일 시스템 및 원격 파일 시스템이 로컬 디스크를 포함하고 있기 때문에 만일 이 값이 7%보다 높으면 디스크가 병목현상을 일으킨다. 디스크가 과부하인지를 판별하기 위해서는 "Disk % Busy"를 관찰하면 된다.

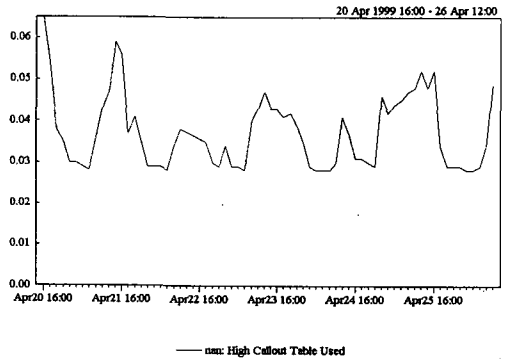


(그림 3) CPU Wait에 의한 이상 분석

(그림 3)은 "CPU Wait"와 "Disk Busy"사이의 관련성을 보여주는 것으로 "Disk Busy"가 "CPU wait"에 영향을 미치고 있음을 알 수 있다.

2.3.3 타이머 이벤트

Callout 테이블은 운영체제에서 동시에 활동할 수 있는 타이머의 수를 제한한 것이다. 타이머는 모든 디바이스 드라이브가 I/O 디바이스의 응답 또는 무응답 결정에 의해 사용된다. Callout 테이블이 오버플로가 되면 시스템은 크래시 된다. 또한 오버플로 되지 않을 경우에도 지속해서 75%가 넘도록 잘못된 프로그램 또는 어플리케이션 그리고 디바이스 I/O를 빈번히 행하는 많은 어플리케이션을 수행할 경우에도 발생한다. 이를 테스트함에 있어 자료의 수집 수를 줄이기 위해 "Callouts % Used"/70;로 하여 0.2 보다 큰 결과를 (그림 4)에 도시하였으며, 실험기간 동안 75%를 초과한 건수를 없었다.



(그림 4) Timer 활용도 측정

3. 성능 시그네쳐 생성

이 연구에서 이상을 탐지의 판별력을 높이기 위한 방법론의 하나로 단일 성능 이벤트를 결합하여 침입 유형에 따른 적절한 성능 시그네쳐 생성 규칙 알고리즘을 제시한다. 이는 각 요소별 프로세스의 유효한 주소 한계 및 시스템 자원의 고갈에 따르는 경계 조건을 기준으로 하여 요소별 연관성을 고려하여 생성한다.

3.1 성능 시그네쳐 생성 알고리즘

성능 시그네쳐 생성 알고리즘은 공격 특성을 기준으로 하여 생성될 수도 있으며, 시스템의 오동작 및 크래시와 관련해서도 생성될 수 있다. 본 연구에서는 후자의 경우를 고려하는데 이는 알려지지 않은 공격 유형에 적절하다. 이에 대한 생성 절차는 (알고리즘 1)과 같다. ClassName은 침입 유형에 적절히 대응하여 시그네쳐를 분류할 수 있도록 하기 위해 정의되고, SELECTION은 시그네쳐를 생성하기 위해 필요한 <표 1>의 시스템 메트릭 아이템을 선정하기 위해 정의된다. 여기서 SELECTION은 시그네쳐 클래스에 따라 단일 또는 복수개를 선택할 수 있다. 테이블에 대한 정의는 단일 메트릭의 위치를 표시하는 것이며, 마지막으로 시그네쳐 클래스에 대한 시그네쳐를 계산한다.

```
Signature_Generator
  {CLASS Class_Name
      SELECTION SelectionName =
        "Process Info.Process<System Metric Item>"
           =value[* - + /];
        :
        :
      TABLE TableName
       COLUMN Type "ColumnName" =
          "Process Info.Process<System Metric Item>";
        :
        :
       [Rate|Kilo|Mega|Total|CPU_Mean];
       ROW RowName Criteria SelectionName;
        :
        :
      end TABLE
    {Signature type "SignatureName" = expression;}
  end CLASS}
end Signature_Generator
```

(알고리즘 1) 성능 시그네쳐 생성 알고리즘

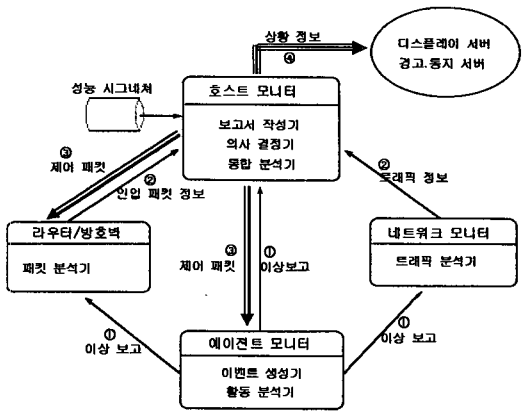
3.2 침입탐지 시나리오

이 장에서는 앞 절에서 설명한 성능 시그네쳐가 어떻게 침입탐지에 유용하게 제공할 수 있는지를 살펴본다. 침입탐지는 성능 시그네쳐를 기준으로 시스템 부하 변화를 분석하여 정상적인 시스템 운영 패턴을 형성한다. 여기에는 네트워크 서비스를 제공하기 위해 미치는 부하도 함께 고려된다. 만일 예외적인 행위가 발생되고 있음을 감지하게 되는데. 이 때 (그림 5)에 도시된바와 같이 에이전트 모니터는 이상에 따른 진단을 호스트 모니터와 네트워크 모니터 그리고 라우터나 방호벽에게 동시에 요청한다. 진단 요청을 받은 각 모니터는 요구 에이전트의 정보를 분석하고 결과를 호스트 모니터에게 제공하며 호스트 모니터는 이를 근거로 침입유무를 분석 및 결정 규칙에 의거 침입 유무를 결정하고, 만일 외부적인 침입(서비스 거부 공격)으로 판단되면, 라우터에게 인입되는 패킷을 드롭 시키거나 적절한 조치를 취할 수 있는 제어 패킷을 발송하게 된다.

앞 절에서 설명한 시그네쳐의 결과와 임계 치를 기준으로 한 탐지 규칙은 다음과 같다.

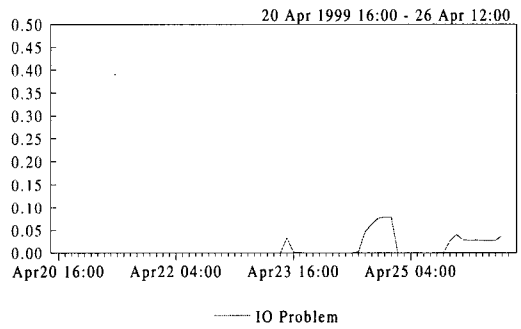
```
IF "SignatureName" Operator value
                    Threshold[SignatureName]
THEN
    Action[message][Second_process][Sleep]
        ...[Kill_Process]
```

여기서 이용되는 연산자는 >, >=, <, <=, 그리고 = 이다.



(그림 5) 침입탐지 시나리오

이를 이용하여 I/O 문제를 야기 시키는 공격을 가정하고 이를 진단하고 관련 이벤트를 확인하는 절차를 실례로 보인다. 이는 CPU의 Idle Time과 Run 큐에 들어 있는 프로세스의 수를 이용하여 이상을 진단하며, 결과 값이 체크 포인트를 넘어서면 현 큐에 있는 프로세스를 조사하여 이상을 야기 시키는 이벤트를 확인한다. 알고리즘 2는 I/O 공격에 대한 이상을 탐지하고 관련 이벤트를 확인하여 조치하는 일련의 프로세스이며, (그림 6)는 성능 시그네쳐를 생성하여 이상을 탐지하는 실험결과를 보인다.



(그림 6) I/O 이상행위 탐지

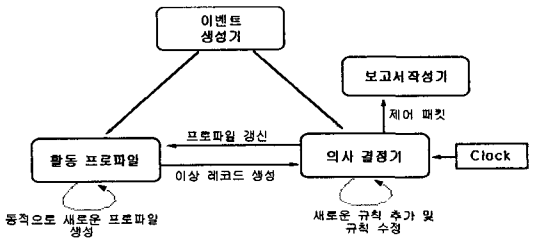
```
/* Signature expression */
  float cpu=min("Cpu % Idle Glob"/40, 1);
  float load=min("Procs Put Run-Q/Sec"/6,1);
  float CP = cpu*load;

  /* Process of Anomaly detect */
  if (CP >= .4)              /* A = Threshold[CP] */
  {
```

```
CHK-Process(newp, max(CPU is idle and Load is high
                    Event_Identification));
print("newp");
report("xcpuproc.rtf");
};
```

(알고리즘 2) I/O 공격탐지 알고리즘

4. 모델 설계

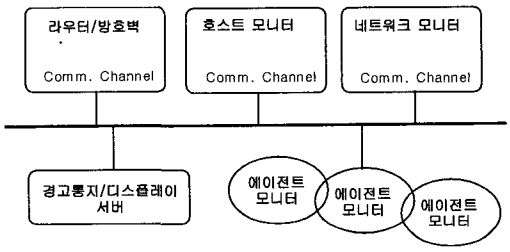
AIDPS의 침입 탐지 모델은 (그림 7)의 기본 모델을 기반[4]으로 하여 침입 탐지 시스템의 주요 논제로 대두되고 있는 자료의 감축과 행위의 효율적 분류 그리고 실시간보고 및 응답 시스템의 프로토타입을 설계한다. 이벤트 생성기는 일반적으로 시스템의 성능 시그네쳐를 기반으로 하는 감사 기록, 네트워크 패킷, 그리고 관측된 다른 행동에 의해 이벤트를 생성한다. 이러한 이벤트는 시스템의 이상을 확인하기 위해 기본으로 제공되며, 활동 프로파일은 침입 탐지기의 전반적인 활동 상태를 나타낸다. 여기에는 미리 학습된 통계적 자료를 이용하여 시스템의 행위를 계산하는 변수들이 포함된다. 또한 패턴 탬플릿을 기반으로 한 새로운 프로필 및 통계 변수 이상 값을 가질 때 이상 레코드를 생성한다. 그리고 규칙 집합은 일반적인 추론 메커니즘을 표현하며, 다른 요소들을 제어하고 상태를 수정하기 위해 이벤트 레코드, 이상 레코드 , 그리고 파기 시간 등을 사용한다. (그림 8)은 AIDPS의 구조로서 5개의 구성요소인 에이전트 모니터, 라우터/방호벽, 호스트 모니터, 네트워크 모니터, 그리고 디스플레이/경고서버로 구성되어 있으며, (그림 5)의 침입탐지 시나리오에 따른 각 구성요소별 기능은 다음과 같다.



(그림 7) 침입탐지 모델

4.1 에이전트 모니터

독립적으로 구동되는 엔티티로서 자신의 시스템의 확실한 상황을 모니터하며, 성능 시그네쳐의 변화가 탐지되면 호스트 모니터에게 보고한다. AIDPS 구조에서 직접적으로 각기 다른 에이전트와 통신을 하지 못한다. 대신, 호스트 모니터에게 그들의 모든 메시지를 보내고 호스트 모니터로부터의 응답 메시지에 의해 어떻게 처리할 것인가를 결정한다. AIDPS 구조는 에이전트의 기능에 대한 요구사항 또는 제한사항을 명시하지 않는다. 이는 특정 시스템 변수 혹은 telnet 연결의 카운팅과 같은 이벤트 또는 국부적인 침입탐지을 위해 복잡한 소프트웨어 시스템을 간단히 프로그램할 수 있음을 의미한다.



(그림 8) AIDPS 구성도

4.2 호스트 모니터

이는 관리 대상 에이전트의 기 정의된 성능 시그네쳐를 유지관리하며, 또한 테스트 프로그램을 구동한 결과를 유지한다. 호스트 모니터는 여러 다른 에이전트 모니터 개체들을 제어하고 적절히 주기적으로 그들을 테스트하여 이상 유무를 점검한다. 모니터는 그들이 제어하는 모든 모니터들이 보고하는 이벤트 정보를 수신하고, 여러 다른 호스트에 포함되어 있는 이벤트를 탐지한다. 또한 수신한 정보를 분석하여 라우터 혹은 해당 에이전트에게 지시 명령을 내린다. 추가적으로, 모니터는 경고통지/디스플레이 서버와 통신하는 능력을 가지며, 전체 AIDPS 시스템에 대한 액세스 포인트를 제공한다.

4.3 라우터/방호벽

각 호스트의 외부 통신 인터페이스로서, 제어와 데이타 프로세싱의 두 가지 역할을 담당하게 되는데, 이는 호스트 모니터에서 제어 패킷을 수신 받아 해당 에이전트 모니터 인입되는 해당 패킷을 패기 하거나 리턴 하는 역할을 수행하며, 명령에 대한 응답으로 적절한 정보를 제공하거나 요구된 행동을 수행한다. 그리

고 호스트 모니터로부터 수신한 명령 처리결과를 다른 에이전트나 호스트 모니터에게 적절히 분배할 수 있다.

4.4 네트워크 모니터

네트워크 모니터는 LAN 내부의 통신량을 분석하여 방어 대상 시스템의 평균 통신량 및 주된 거래 IP 리스트 그리고 전체 평균 통신 부하 분포도를 유지 관리한다. 에이전트 모니터의 이상보고 접수시 해당 에이전트에 대한 현재 통신량(요구량)과 평균 통신량 그리고 전체 부하량을 호스트 모니터에게 통보한다. 네트워크 모니터는 이를 분석하여 자체 진단 결과를 통보할 수도 있다. 이는 보편적으로 NMS (Network Management System)을 이용하여 구현될 수 있다.

4.5 경고통지/디스플레이 서버

AIDPS구조는 데이타 수집과 프로세싱 요소로부터 사용자 인터페이스를 명확히 분리하여 사용자 인터페이스는 호스트 모니터와 상호작용을 해야하고, 호스트 모니터가 정보를 요구하고 명령을 제공하기 위해 제어 스크립터를 이용한다. 이 구분은 AIDPS 시스템과 함께 사용될 수 있는 다른 사용자 인터페이스 구현을 허용한다.

5. 결 론

이 논문에서는 시스템 시큐리티 위반을 탐지하는데 이용되는 기법에 성능 시그네쳐 정보를 추가할 수 있는지를 각 시스템들이 제공하고 있는 측정 메트릭들을 조사하여 실험 분석하였고, 이를 기반으로 성능 시그네쳐를 생성하는 알고리즘을 제시하고 검증하였다. 그리고 생성된 성능 시그네쳐를 이용하여 서비스 거부 공격에 대응할 수 있는 협력적인 서비스 거부 침입탐지 모델을 제시하였다. 앞으로의 주된 연구 방향은 간접적인 서비스 거부 공격의 원인을 진단할 수 있는 성능 시그네쳐 생성 기법을 연구하여 시스템에 추가하는 것이며, 이 기법의 효율성을 최대화하도록 하는 것이다. 비록 측정되는 가능한 많은 성능 메트릭을 형성할 수 있지만, 많은 소프트웨어에 부하를 주지 않고 침입을 잘 탐지할 수 있는 집합을 결정해야한다. 또한 이 접근법의 논리적인 스텝은 바람직하지 못한 프로그램 또는 시스템 행위를 확인과 더불어 정상적인 성능 시그네쳐를 정확히 확인하여 이상 발생시 그 원인을 집중하하고, 그에 대한 적절한 반응을 나타내도록 하는 것이다. 따라서 시스템 운영의 정상적인 패턴을 확실히 확인하기 위하여 소프트웨어 성능 분석을 위해 초기에 개발된 기법을 조사하며, 각종 서비스 거부 공격의 유형이 시스템 구성 요소에 미치는 영향을 분석하고 체계화하여 성능 시그네쳐의 판별력을 보다 강화할 수 있도록 할 것이다.

참 고 문 헌

[1] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff. "A sense of self for Unix processes," Proceedings of the 1996 IEEE Symposium on Computer Security and Pri-vacy, 1996.

[2] Teresa F. Lunt. "Detecting intruders in computer systems," 1993 Conference on Auditing and Computer Technology, 1993.

[3] Karl Levitt, Calvin Ko, and George Fink. "Automated detection of vulnerabilities in privileged programs by execution monitoring," 1994 Computer Security Application Conference, 1994.

[4] Dorothy E. Denning. "An intrusion detection model," IEEE Transactions on Software Engineering, vol.SE-13, no.2, Feb. 1987, pp.222-232.

[5] Debra Anderson, Teresa F. Lunt, Harold Javitz, Ann Tamaru, and Alfonso Valdes. "Detecting unusual program behavior using the statistical component of the Next Generation Intrusion Detection Expert System(NIDES)".

[6] K. Ilgum, R.A. Kemmerer, and P.A. Porras, "State Transition Analysis: A Rule-Based Intrusion Detection Approach," IEEE Transaction on Software Engineering, Vol.21, No.3, March 1995, pp.181-199.

[7] Stuart Staniford-Chen. "Common intrusion detection framework," WWW page at http://seclab.cs.ucdavis.edu/cidf/.

[8] "정보시스템 해킹현황 및 대응", 한국정보보호 센터, 1996. 11.

[9] "침입 여부 판정 엔진 모듈 설계 및 프로토타입 개발", 한국정보보호센터, 1997. 12.

[10] "침입 탐지 모델 분석 및 설계", 한국정보보 호센터, 1996. 12.

김 광 득

e-mail : kdkim@kier.re.kr

1987년 대전산업대학교 전자계산학과 졸업(공학사)

1989년 전북대학교 대학원 전산통계학과 졸업(이학석사)

1994년~현재 충북대학교 대학원 전자계산학과 박사과정 수료

1981년~현재 한국에너지기술연구소 선임기술원

관심분야 : 컴퓨터 보안, 침입탐지 시스템, 네트워크 관리 등

이 상 호

e-mail : shlee@cbucc.chungbuk.ac.kr

1976년 숭실대학교 전자계산학과 졸업(공학사)

1981년 숭실대학교 대학원 전자계산학과 졸업(공학석사)

1989년 숭실대학교 대학원 전자계산학과 졸업(공학박사)

1976년~1979년 한국전력 전자계산소 프로그래머

1981년~1983년 한국전자통신연구소 위촉연구원

1990년~1991년 호주 텔레콤 연구소 방문연구원

1992년~1993년 캐나다 UBC 방문연구원

1981년~현재 충북대학교 컴퓨터과학과 교수

관심분야 : Protocol Engineering, Network, Security, Network Management, Network Architecture 등