침입기법별 침입 탐지 알고리즘과 구현

코코넛(주) 김주영
니츠(주) 강창구
한남대학교 소우영 · 이일근 · 이 극*

1. 서 론

인터넷을 통한 전자상거래, 홈뱅킹 등 네트워크를 이용한 새로운 서비스들이 다양하게 개발되어 사용자가 크게 증가되고, 인트라넷 구축을 통한 공, 사기업에서 전자기록 이용이 보편화됨에 따라 정보화사회로의 변화가 급속하게 진행되고 있다. 그러나 이를 악용하는 불건전정보 유통 및 정보범죄와 같은 정보화의 역기능 또한 크게 증가하고 있다. 정보범죄의 유형은 전산망 침해행위, 전자기록 위·변조, 각종 음란물 유통, 통신상의 명예훼손, 바이러스 제작 유포 등이 있으며, 특히 해킹과 같은 대표적인 전산망 침해행위는 적절한 예방과 단속대책이 필요하다.

일반적으로 침입자가 컴퓨터 시스템에 침입하는 과정은 3단계로 구분하는데, 대규모 네트워크나 특정 호스트의 취약점을 검색하는 네트워크 취약점 검색단계와 목적 호스트로 접근하여 사용자나 관리자의 권한을 획득하는 권한획득단계, 마지막으로 획득한 권한을 재사용하기 위해서 관리자로부터 자신을 숨기기 위한 백도어(back door) 설치단계로 나눌 수 있다. 이러한 침입위협에 대처하기 위해 침입을 탐지하기 위한 기술인 침입탐지시스템에 관한 연구가 활발히 진행되고 있다.

침입탐지시스템은 데이터의 소스를 기반으로 하는 분류 방법과 모델을 기반으로 하는 분류 방법이 있다.

데이터 소스를 기반으로 한 방법은 호스트 기반, 다중 호스트 기반, 네트워크 기반의 침입탐지 기반으로 분류한다. 호스트 기반 침입탐지 시스템은 단일 호스트로부터 생성되고 수집된 감사자료(Audit Data)를 침입여부 판정에 사용하며, 하나의 호스트를 탐지영역으로 한다. 멀티호스트 기반 침입탐지 시스템은 여러 호스트들로부터 생성되고 수집된 감사자료를 침입 여부판정에 사용하며, 여러 대의 호스트를 그 탐지영역으로 하기 때문에 호스트간의 통신을 통해 침입 판정에 필요한 정보를 교환하게 된다. 네트워크 기반 침입탐지 시스템은 네트워크의 패킷을 수집하여 침입여부판정에 사용하며, 침입탐지시스템이 설치된 네트워크의 영역 전체를 탐지 대상으로 할 수 있다. 네트워크의 패킷 처리 능력이 침입탐지시스템의 성능을 결정짓게 되므로 성능에 적합한 네트워크를 대상으로 하여야 한다.

모델을 기반으로 하는 분류 방법은 비정상 침입탐지기법(anomaly detection)과 오용탐지기법(misuse detection)으로 분류한다. 비정상 침입탐지 기법은 사용자의 패턴을 분석하여 입력패턴과 비교하여 침입을 탐지한다. 비정상 침입탐지기법의 종류는 통계적인 방법(Statistical approaches), 예측가능한 패턴 생성(Predictive Pattern generation), 신경망(Neural Network), 특징추출(Feature Selection) 등이 있다. 오용 침입탐지 기법은 알려진 침입행위를 이용하여 침입을 탐지한다. 오용 침입탐지의 종류는 전문가 시스템(Expert Systems), 모델에 근거한 침입탐지(Model based Intrusion Detection), 상태전이

* 종신회원

분석(State Transition Analysis), 패트리넷(Petri-Net)을 이용한 방법 등이 있다.

일반적으로 침입탐지시스템은 시스템 관리자의 전문적인 지식에 의존하지 않고 지속적으로 수행되어야 하며, 컴퓨터 시스템에 최소한의 오버헤드를 부과해야 하고, 정상상태를 침입으로 탐지하는 거짓 탐지율(false positive)과 침입상태를 정상상태로 판단하는 탐지 실패율(false negative)을 최소화해야 한다.

이와 같은 침입탐지시스템을 개발하기 위해 다양한 모델들이 개발되고 있으나 아직 외국의 기술을 따라 가고 있거나 완벽하지 못한 모델을 개발하고 있다. 또한 개발된 시스템은 고가이어서 시스템의 공개 및 기술이전을 꺼리는 실정이므로 정보보호에 대한 중요성이 증대되고 있는 시점에서 침입탐지시스템 개발을 위한 지속적인 노력이 필요하다. 특히 네트워크 기반 침입탐지시스템은 호스트 기반 침입탐지시스템보다 빠른 탐지를 하여 실시간 처리가 가능하며, 시스템 내부로의 침입을 탐지하는 것은 물론 침입을 시도했으나 실패한 경우에도 탐지가 가능하다.

본 논문에서는 네트워크에서 수집한 패킷을 분석하여 외부로부터의 침입을 사전에 탐지할 수 있는 네트워크 기반 침입탐지시스템을 설계하고 구현하며, 침입유형별로 분류된 탐지엔진을 위한 탐지알고리즘을 제시한다.

2. 탐지기법 분석

해커가 컴퓨터에 침입하는 과정을 네트워크 취약점 검색단계, 사용자 권한 획득단계, 백도어 설치단계로 나누었을 때, 각각 복잡형, 단순형, 지능형에서 이를 탐지할 수 있다. 복잡형에서는 시간과 횟수에 대한 임계치를 설정하여 네트워크의 취약점 검색공격에 대한 탐지를 할 수 있으며, 단순형에서는 사용자 권한 획득이나 서비스 거부 공격을 위한 탐지를 수행한다. 마지막으로 지능형 탐지에서는 백도어를 심는 과정을 룰베이스로 정의하여 TCP 서비스를 이용한 침입자의 명령어를 분석하여 일치하는 부분을 탐지한다.

2.1 단순형 탐지기법

단순형 탐지 기법은 저수준 필터링에서 수집된 감사자료를 침입탐지에 사용한다[7, 10].

2.1.1 Source Routing

Source Routing의 기본 개념은 송신자가 라우터를 지정하는 것이다. IP 헤더의 option field를 조작하여 라우팅 경로를 지정하여서 패킷을 전송하면 그에 대한 응답은 지정된 경로의 역순으로 전송된다. 공격자는 자신의 IP 주소를 신뢰받는 호스트의 IP로 바꾸고 source route를 설정한 후 rlogin이나 telnet으로 접속을 시도하면 된다[10].

```
procedure SourceRouting_Detection
Detect Protocol
if(IP_header_option == IPOPT_SSRR)
/* Strict Source Routing Option이 Set 확인 */
if(IP_header_size > Upper_threshold ||
  IP_header_size < Lower_threshold)
then alarm "Source Routing Detected"
```

2.1.2 Smurf Attack

Ping이라는 프로그램으로 잘 알려진 ICMP 프로토콜의 echo request/reply 패킷은 서비스 거부인 Smurf 공격으로 악용될 수 있다. 공격하고자 하는 Victim의 IP주소를 source address로 하여 ICMP echo request를 local network 전체에게 브로드캐스트 주소로 보내게 되면 Victim의 호스트는 되돌아오는 echo reply 패킷 때문에 마비상태에 빠지게 된다. Smurf 공격은 ICMP echo request 패킷 중에서 브로드캐스트 주소를 가진 패킷을 필터링하거나, 임계치 이상의 echo reply 패킷이 도착하는 것을 카운트하는 방법으로도 탐지할 수 있다[5, 7]. 후자의 방법으로는 5초안에 5번 이상의 echo reply를 탐지하게 된다[11].

```
procedure SmurfAttack_Detection
if(ICMP_Message_Type == echo request)
      if(Audit_Data->Destination_Address ==
 BROADCAST_Address)
        then alarm "Smurf Attack Detected"
```

2.1.3 Land Attack

```
procedure LandAttack_Detection
if (Audit_Data->Source_Address ==
 Audit_Data->Destination_Address)
        then alarm "Land Attack Detected"
```

Land 공격은 소스 시스템의 IP 주소를 공격하고자 하는 시스템의 IP 주소와 포트로 바꾸어 전송하면 공격받은 호스트는 루프상태에 빠져 IP 스택에 심각한 장애를 유발하게 된다.

Land 공격을 탐지하기 위해서는 라우터에서 내부 어드레스 주소를 가진 외부 패킷을 필터링하거나 소스 주소와 대상 주소가 같은 경우를 비교하면 된다.

2.2 복잡형 탐지기법

복잡형 탐지 기법은 그림 1과 같이 일정한 시간 내에(Time_threshold) 연속된 연결요청(ConReq_Count)이 임계치(ConReq_Threshold)와 같게 되는 경우를 검사하여 탐지를 하게 된다[2].

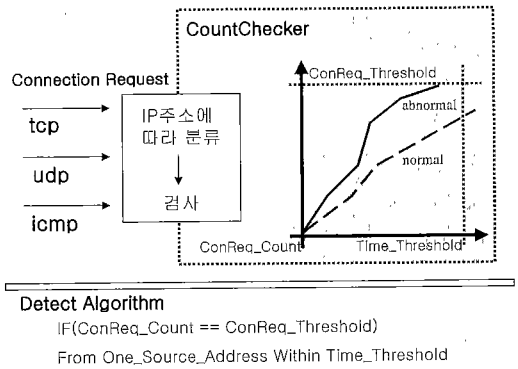


그림 1 복잡형 탐지기법

2.2.1 네트워크 검색 공격

네트워크 검색 공격은 직접적인 공격의 방법이라고 할 수는 없지만, 대상 호스트의 취약점을 검색할 수 있기 때문에 침입의 전 단계로 사용되고 있다. 네트워크 검색 공격을 탐지하기 위해서 감사자료 수집모듈에서 수집한 패킷 중에서 TCP 패킷을 필터링하고 그중 연결요청 패킷인 SYN 패킷을 필터링하여 TCP로 연결요청을 하는 패킷만을 수집한다. 주어진 일정한 시간 안에 하나의 IP 소스로부터 임계치 이상의 연결 요청이 발생하면 침입으로 판정한다. 연결 요청은 TCP 헤더의 13번째 배열을 2로 mask bit 연산을 하면 SYN 패킷만 추출할 수 있다. 대부분의 경우에 하나의 IP 주소로부터 추출된 연결요청 패킷을 카운트하여 5~7초안에 7번 이상의 연결요청이 발생하면 네트워크 스캔공격으로 탐지하게 된다. 그러나 WWW에서 하나의 웹페이지로부터 다수의 gif 그림 파일을 서로 다른 URL로부터 다운로드 받도록 설정해 놓은 경우가 있는데, 이러한 경우에는 각 그림을 다운로드 받기 위해서 7번 이상의 연결요청이 발생하게 된다. 따라서 WWW 서비스의 경우에는 연결요청 임계치를 40으로 설정하여 이러한 거짓 탐지율(false positive)를 최소화하여야 한다[11].

```
procedure SourceRouting_Detection
integer maxtimes
integer duration
if TCP[13] & 2  =! 0 /* 연결 요청 패킷 */
    Count_packet(maxtimes-1, time()+duration,
Audit_Data->Destination_address)
Count_packet(countdown, timestamp, ip, port)
if (TCP[13] & 2 =! 0) and
(Audit_Data->Destination_Address == ip) {
    if countdown > 1
        Count_packet(countdown-1, timestamp, ip)
        else if countdown == 1
                alarm "abnormal"
    }
if time() >= timestamp
break;
```

2.2.2 SYN flooding 공격

TCP 프로토콜에서 서버와 클라이언트의 연결은 3-Way Handshaking에 의해서 이루어지는데, 클라이언트의 SYN 플래그를 설정하여 서버에게 연결시도 후에, 서버의 SYN-ACK 플래그를 설정한 패킷을 클라이언트에게 보내는 과정에서 서버는 half-open 상태로 머물게 된다. 만약 클라이언트가 서버의 SYN 응답에 ACK를 보내지 않으면 TCP 프로토콜은 이에 대한 대책이 정의되어 있지 않기 때문에 TCP 연결을 위해서 backlog queue에 연결 상태를 유지하기 위한 정보를 저장하고 half-open 상태로 계속 머무르다가 일정 시간이 지나도 ACK 패킷이 도착하지 않으면 다시 정상적인 상태로 복귀하게 된다. 그러나 이러한 일이 계속적으로 발생하게 되면 시스템이 일정시간 후에 백로그 큐에서 연결정보를 삭제하는 것 보다 더 빠르게 일어나 백로그 큐가 가득 차게 되어 이후에 이 포트에 대한 TCP 연결요청을 모두 거부할 수밖에 없게 된다.

TCP의 연결요청 패킷을 검사하여 일정한 시간 안에 하나의 소스로부터 다수의 연결요청이 발생하게 되는 것을 탐지한다. SYN flooding과 네트워크 스캔공격은 하나의 IP 주소로부터 연결요청을 시도하는 횟수를 카운트하는 것은 동일하나, 네트워크 스캔공격이 여러 포트를 대상으로 하는 반면에 SYN flooding 공격은 하나의 포트를 대상으로 하기 때문에 하나의 소스로부터 하나의 포트로 임계치 이상의 연결요청을 카운트하게 된다[11].

```
procedure SourceRouting_Detection
integer maxtimes
integer duration
if TCP[13] & 2  =! 0
    Count_packet(maxtimes-1, time()+duration,
Audit_Data->Destination_address,
Audit_Data->Port_Num)

Count_packet(countdown, timestamp, ip, port)
if (TCP[13] & 2 =! 0) and
(Audit_Data->Destination_Address == ip) and
(Audit_Data->Port_Num == port) {
    if countdown > 1
    Count_packet(countdown-1, timestamp, ip, port)
    else if countdown == 1
        alarm "abnormal"
}
if time() >= timestamp
    break;
```

2.3 지능형 탐지기법

불법 침입자가 침입에 성공하면 시스템 접근에 대한 사용자 인증 등 정상적인 절차를 거치지 않고 응용 프로그램 또는 시스템에 접근할 수 있도록 back door를 심게 된다. 지능형 탐지에서는 그림 2와 같이 규칙기반 데이터베이스에 있는사전에 분석된 침입의 유형에 대해 시스템에 접속하는 모든 telnet 로깅정보와 비교하여 침입을 탐지하게 된다[3, 5, 7].

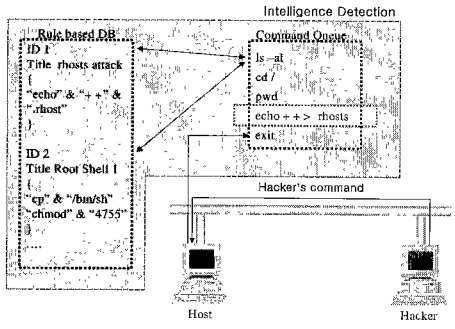


그림 2 지능형 탐지기법

2.3.1 버퍼 오버플로우

메모리의 구조는 프로그램이 메모리 상에서 수행될 때 프로그램이 들어가는 자리인 Text 부분과 이미 설정되어 있는 data, static 변수 등의 데이터가 들어 있는 Data 부분, 임시로 기억되어질 내용을 담는 stack 영역으로 분류된다. 이 때, main()과 서브 함수인 function()이 있다고 할 때, function()에 있는 로컬 변수를 오버 플로우 시켜 return address에 있는 원래의 복귀 주소를 변경하여 루트 권한의 쉘코드를 실행시킨다. 만약 실행 시에 루트권한을 갖는 Set User ID 비트가 설정된 프로그램에 오버 플로우 취약점이 있다면, 쉘코드를 실행시키지 않고도 루트권한을 획득할 수 있다.

버퍼오버플로우를 이용한 공격에 사용되는 프로그램에는 쉘코드가 삽입되는데, 쉘을 실행시키는 프로그램을 Static 방식으로 컴파일하고 난후에, 실행파일을 gdb(gcc Debugger)란 유틸리티를 사용하여 쉘을 수행하기 위한 함수를 호출하는 __execve() 부분을 disassemble하면 System Call을 부르는 Library Routine을 알 수 있다. 이 코드를 이진코드 형태로 바꾼 후에 이것을 지역 변수의 영역에 삽입해 주면 시스템 프로그램 속에 쉘의 코드가 삽입되게 된다. 쉘코드는 운영체제, CPU에 따라 다른 코드를 갖게 되며 x86 계열의 Linux CPU에서는 다음과 같다.

```
char shellcode[] =
"\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46"
"\x07\x89\x46\x0c\xb0\x0b\x89\xf3\x8d\x4e"
"\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8"
"\x40\xcd\x80\xe8\xdc\xff\xff\xff/bin/sh";

void main() {
        int *ret;
        ret = (int *)&ret + 2;
        (*ret) = (int)shellcode;
}
```

위 프로그램을 수행시키면 쉘이 수행되는 것을 볼 수 있는데, SETUID를 가진 취약점이 있는 프로그램의 이름을 Vulnerable이라고 할 때, 아래와 같이 추가시켜 수행시키면 루트권한의 쉘을 얻을 수가 있다.

```
#include <stdio.h>
#include <stdlib.h>
char shellcode[] =
"\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46"
"\x07\x89\x46\x0c\xb0\x0b\x89\xf3\x8d\x4e"
"\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8"
"\x40\xcd\x80\xe8\xdc\xff\xff\xff/bin/sh";

unsigned long get_sp(void) {
        __asm__("movl %esp, %eax");
}

void main(int argc, char *argv[]) {
        char buff[1133], *ptr;
        long *addr_ptr;
        int i, offset;
        for(i=0;i<1133;i++)
                buff[i] = 0x90;
        if(argc>1)
                offset=atoi(argv[1]);
        ptr=buff;
        addr_ptr=(long *)ptr;
        *(addr_ptr+282)=get_sp()-offset;
        ptr=buff+1083;
        for(i=0;i<45;i++)
                *(ptr++)=shellcode[i];
        buff[1132]='\0';
        execl("./Vulnerable", "Vulnerable", buff,
NULL);
}
```

이러한 버퍼오버플로우를 탐지하기 위해서는 취약점을 검사할 수 있는 프로그램을 아래와 같이 작성한다. 아규먼트의 값을 프로그램이 수행시에 검사를 하여 정상적인 아규먼트이면 원래의 프로그램을 수행시키고, 비정상적인 패킷이 입력되면 탐지 및 대응을 하게 된다.

```
#define MAXARGLEN 16
#define REAL_PROG "Buffer Overflow 취약점이
있는 파일"
procedure BufferOverflowDetect(int argc,
char *argv[], char *envp[]) {
int i;
for(i=0; i<argc; i++) {
 if(strlen(argv[i]) > MAXARGLEN) {
        alarm "BufferOverflow Detected"
   }
 }
 execve(REAL_PROG, argv, envp);
 /* 아규먼트의 길이가 허용되면 프로그램 실행
*/
 }
```

2.3.2 Rule base에 정의된 공격

불법 침입자가 침입에 성공하면 시스템 접근에 대한 사용자 인증 등 정상적인 절차를 거치지 않고 응용 프로그램 또는 시스템에 접근할 수 있도록 back door를 심게 된다. 시스템으로 접속하는 모든 사용자의 telnet 서비스 이용 시에 타이핑하는 명령어, 아규먼트, 옵션의 정보를 기록하여 분석하면, 보다 완벽한 네트워크 보안을 이룰 수 있다. 지능형 탐지에서는 이러한 침입의 유형을 분석하여 룰베이스에 저장하게 되며, 시스템으로 접속하는 모든 telnet 로깅정보를 분석하여 정의된 공격의 유형을 탐지한다. 다음은 유닉스 시스템에서 환경변수(Environment Variable)를 이용한 공격중 하나인 IFS(Internal Field Separator) 공격을 탐지하는 예이다.

```
/* return값이 1 : 침입패턴 일치, 0, -1 : 패턴
불일치 */
int Rule4() {
char* sp =
 strstr(CommandQueue[rear].command, "IFS");
if( sp == NULL ) return 0;
sp = strstr(sp, "=")
if( sp == NULL ) return 0;
sp = strstr(sp, "\"/\"");
if( sp == NULL ) return 0;
if( front == rear+1 ) return -1;
sp =
strstr(CommandQueue[rear+1].command, "export");
if( sp == NULL ) return 0;
sp = strstr(sp, "IFS");
if( sp == NULL ) return 0;
return 1;
}
```

CommandQueue는 접속한 사용자의 모든 명령어가 들어가 있는 구조체이다. Command Queue 구조체의 형식은 다음과 같다.

```
struct commandQueue{
 long SourceAddr, DestAddr;
  /* 소스와 목적지 IP 주소 */
 int SourcePort, DestPort;
  /* 소스와 목적지 포트번호 */
 char command[MAX_CHAR];
  /* 명령어가 들어있는 큐 */
}CommandQueue[MAX_QUEUE_SIZE];
```

3. 네트워크 기반 침입탐지 시스템 설계 및 구현

본 장에서는 2장에서 제안된 탐지알고리즘을 이용하여 전체적인 네트워크 기반 침입탐지시스템을 설계한다. 2장에서 단순형, 복잡형, 지능형 탐지로 분류한 침입탐지엔진 모듈 외에 네트워크로부터 감사자료(Audit Data)를 수집하는 감사자료 수집모듈과 탐지된 결과를 관리자에게 보고하고 침입자에게 대응하는 침입보고 및 대응모듈, 등으로 나눌 수 있다.

3.1 개발 환경

시스템을 개발하기 위한 소프트웨어와 하드웨어 환경은 다음과 같다.

System : Intel Pentium 350MHz
OS : Linux 6.0
Packet Filter : Libpcap ver 4.0
Language : C/C++ Language
Compiler : gcc, g++ Compiler

3.2 시스템의 구성

네트워크 침입탐지 시스템의 구성은 그림 3과 같다. 감사자료 수집모듈은 저수준 필터링과 고수준 필터링으로 나누어 패킷을 수집한다. 저수준 필터링에서는 이더넷 프레임 헤더, TCP 헤더, IP 헤더, UDP 헤더 등에서 얻을 수 있는 감사자료를 필터링하고, 고수준 필터링에서는 TCP 세그먼트의 사용자 데이터 부분에서 얻을 수 있는 명령어, 아규먼트 등을 어플리케이션 수준으로 감사자료를 필터링한다.

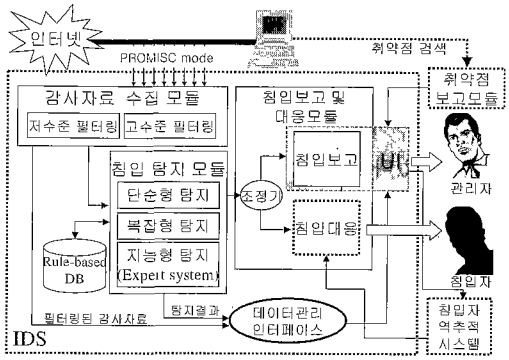


그림 3 제안된 네트워크 침입탐지 시스템 모델

침입탐지 모듈은 단순형, 복잡형, 지능형으로 나누어 침입을 탐지한다. 단순형과 복잡형에서는 저수준 필터링에서 수집된 감사자료를 바탕으로 침입을 탐지하며, 지능형 탐지는 고수준 필터링에서 수집된 것을 사용한다.

침입보고 및 대응모듈에서는 침입탐지 모듈에서 탐지한 결과를 관리자에게 콘솔, 이메일, 정보통신기기 등으로 보고하게 되며, 침입자에게는 침입의 유형에 따라 대응을 하게 된다.

취약점 보고모듈은 침입탐지 시스템이 설치되기 전에 호스트의 취약점을 관리자에게 보고하는 모듈이다.

3.3 감사자료 수집 모듈

감사자료 수집모듈은 libpcap-0.4 라이브러리를 이용하여 구현하였다. libpcap은 UC Berkeley에서 개발한 패킷 수집을 효과적으로 할 수 있게 만든 공용 라이브러리이다. 다음은 libpcap을 이용하기 위한 인터페이스 부분을 구현한 예이다.

libpcap 라이브러리를 이용하여 수집된 패킷은 그림 4와 같이 저수준 필터링과 고수준 필터링 방법으로 패킷을 축약하게 된다[4].

```
#include<pcap.h>
...
device = pcap_lookupdev(buf);
/* 가상 디바이스의 이름 확인 */
pd = pcap_open_live(device, snaplen, !pflag, 1000,
ebuf);
/* 가상 디바이스 open,
 !pflag 가 1이면 promiscuous 모드 */
pcap_lookupnet(device, &localnet, &netmask, ebuf);
```

```
/* 현재 네트워크의 netmask 확인 */
pcap_compile(pd, &fcode, cmdbuf, Oflag, netmask);
/* 패킷 필터 컴파일 */
pcap_setfilter(pd, &fcode)
/* 컴파일 된 패킷 필터를 가상 디바이스에 설정 */
pcap_loop(pd, cnt, printer, pcap_userdata)
/* 패킷 read */
...
```

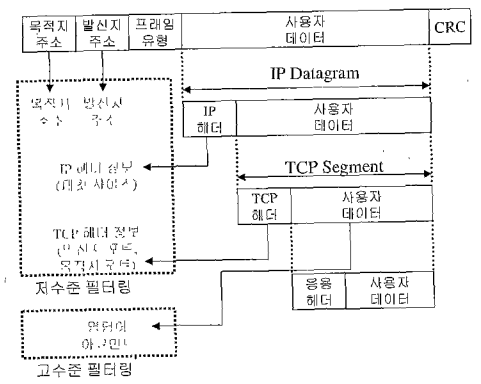


그림 4 패킷에서 필터링 과정

패킷의 수집된 내용을 고수준 필터링 방법과 저수준 필터링 방법에 따라 분류한 데이터의 종류는 표 1과 같다.

표 1 필터링된 데이터의 종류와 내용

	구성요소	설 명
저수준 필터링	Net_Ser	전산망 서비스
	Src_Sys	서비스를 요청한 시스템
	Dest_Sys	서비스의 목적 시스템
	Src_Addr	소스 시스템의 IP 주소
	Dest_Addr	목적 시스템의 IP 주소
	Error_Cond	에러 상태값
	SerReq_Time	서비스를 요청한 시간
	SerFin_Time	서비스를 끝낸 시간
	Port_Num	포트 번호
고수준 필터링	Cmd	명령어
	Arg	아규먼트
	Opt	명령어의 옵션
	User	서비스를 요청한 사용자

3.4 침입탐지모듈

침입탐지 모듈은 수집된 네트워크 패킷을 이용하여 단순형, 복잡형, 지능형으로 나누어 탐지하게 된다.

3.5 침입보고 및 대응모듈

네트워크에서 수집된 감사자료를 이용하여 침입판정모듈에서 침입이 탐지되면 관리자에게 침입의 결과를 보고하게 되며 침입의 유형에 따라 침입자에게 대응을 하게 된다. 침입보고는 시스템의 콘솔, 전자우편, 정보통신기기 등을 이용하여 침입의 사실을 보고하게 된다.

침입자를 위한 침입대응모듈은 침입의 유형에 따라 침입자에게 전자우편을 보내거나, 시스템에서 로그아웃 시키는 방법이 있다. 또한 서비스 거부와 같이 대상시스템이 공격을 받은 후 심각한 상황에 빠질 수 있는 경우에는 시스템의 네트워크 인터페이스를 단절시켜 침입시도를 막게 된다.

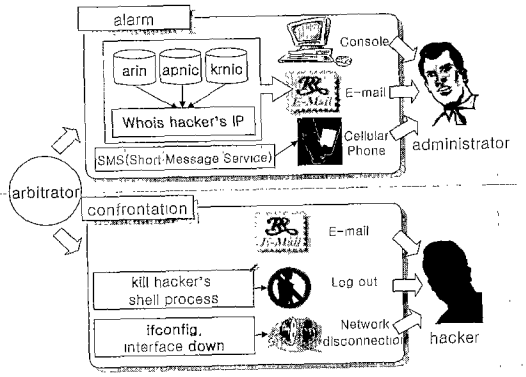


그림 5 침입보고 및 대응모듈의 구성

3.6 취약점 보고모듈

유닉스 기반의 취약점 자동 검색 및 보고 모듈은 그림 6과 같은 순서도를 이용하여 시스템의 취약점을 검색하여 결과를 관리자에게 보고한다. 취약점이 있는 소프트웨어의 버전정보를 데이터베이스로 구축한 후에 설치된 소프트웨어의 버전과 비교하여 일치하는 버전을 가진 소프트웨어의 리스트를 작성하여 관리자에게 보고하게 된다.

4. 성능 평가

제안된 네트워크 침입탐지 알고리즘과 시스템을 평가하기 위해서, 네트워크 스캔 공격과 사용

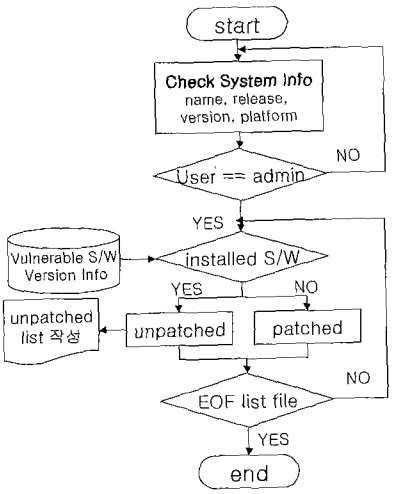


그림 6 취약점 보고모듈의 순서도

자의 명령어를 분석하기 위한 비정상적인 명령어를 탐지하는 침입유형을 탐지했을 때 실시간 침입탐지를 할 수 있는 타당성을 살펴본다.

침입탐지를 위한 시도된 침입의 기법은 4가지이다. 첫 번째와 두 번째는 mscan과 sscan을 이용한 네트워크 스캔 공격이다. 세 번째와 네 번째는 임의의 사용자의 비정상적인 명령어를 검사하는 지능형 탐지방법이다. 세 번째는 사용자가 "echo ++ > .rhosts" 이란 명령어를 수행하였을 경우이며 네 번째는 "cp /bin/sh, chmod 4755" 란 명령어를 수행하여 루트권한의 쉘을 복사하는 경우이다.

표 2는 각각의 침입기법에 대해서 일차적으로 탐지한 시간들, 관리자에게 보고를 위해 메일과 휴대폰으로 보고된 시간을 기록하였다. 일차적인 탐지시간과 메일로 보고된 시간을 살펴보면 실시간 처리가 가능하다. 이에 비해 휴대폰으로 호출하는 시간은 SMS(Short Message Service)를 이용하는 시간이 소요되므로 다소 지연되지만 관

표 2 침입탐지 평가 1

(단위 : sec)

조건 / 침입기법	탐지시간	보고시간	
		메일	휴대폰
침입기법 1	0.5	1.0	15.5
침입기법 2	0.8	1.1	14.7
침입기법 3	1.2	1.5	15.8
침입기법 4	1.3	1.8	14.0

표 3 침입탐지 평가 2

조건 / 침입기법	거짓탐지율 발생횟수	탐지실패율 발생횟수
침입기법 1	1	0
침입기법 2	1	0
침입기법 3	0	0
침입기법 4	0	0

리자가 항상 메일이나 콘솔로 보고 받을 수 있는 상황에 있지 않는 부재중인 경우를 대비하여 꼭 필요하다.

표 3은 침입탐지시스템에서 성능평가의 중요한 요소로 작용하는 거짓탐지율과 탐지실패율을 각각의 침입기법에 따라 비교하였다.

위의 자료에 따르면 침입이 아닌 경우에 침입이라고 탐지하는 거짓탐지율이 1,2번 기법에서 한 번씩 발생하였으나, 침입을 탐지하지 못하는 탐지실패율은 모든 경우에서 발생하지 않았다.

5. 결 론

본 논문에서는 침입을 탐지할 수 있는 기법들을 침입기법별로 분석하고 알고리즘을 제안하였다. 또한 이를 제안된 알고리즘을 적용하여 네트워크 기반 침입탐지시스템을 설계하고 구현하여 실시간 네트워크 기반 침입탐지 시스템을 위한 타당성을 보였다.

네트워크 침입탐지 기법은 IP 패킷 분석을 하기 때문에 호스트 기반의 침입 탐지기법보다 빠른 실시간 처리를 가능하게 하며, 침입을 시도했으나 실패한 침입의 경우에도 탐지할 수 있는 장점이 있다. 또한 해커들의 침입이 성공한 후에는 접속한 흔적을 지우기 위해서 자동으로 로그파일을 지워주는 프로그램을 사용하므로, 로그정보만으로 탐지하기에는 한계가 있다. 그러므로 네트워크 기반 침입탐지 기법은 보다 철저한 네트워크 보안관리를 위해서 중요하다.

네트워크 패킷은 계속 수집되기 때문에 탐지 알고리즘의 처리가 늦어지게 되면 연속해서 들어오는 패킷의 손실이 발생하게 된다. 따라서 거짓탐지율과 탐지실패율을 최소화하는 범위 내에서 탐지 알고리즘을 단순화할 필요가 있다. 또한, 새로운 침입 유형에 대한 탐지방법은 계속 연구되어야 할 과제이다.

참고문헌

[1] 이경하 외 3명, "네트워크 패킷 정보를 기반으로 한 보안 관리", 한국정보과학회 논문지, Vol. 25, No.12, pp.1405-1412, Dec. 1998.

[2] 김기중 외 3명, "유형별 침입자 감지를 위한 감사추적 및 분석 시스템 모델", 한국정보과학회 논문지, Vol 25, No.2, Feb 1999.

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

[4] Steven McCanne, Van Jacobson, "The BSD Packet Filter: A New Architecture for User-level Packet Capture", December 19, 1992.

[5] Sandeep Kumar, "An Application of Pattern Matching in Intrusion Detection", Technical Report CSD-TR-94-013 Coast TR 94-07, June 17, 1994.

[6] Eugene H. Spafford, "A Software Architecture to support Misuse Intrusion Detection" Technical Report CSD-TR-95-009 Coast TR 95-04, March 17, 1995.

[7] Vern Paxson, "Bro: A System for Detecting Network Intruders in Real-Time", Jan 14, 1998.

[8] http://www.certcc.or.kr

[9] http://www.rootshell.com

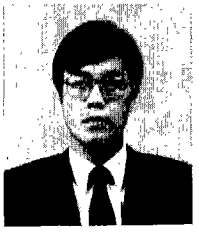
[10] W. Richard Stevens, Unix Network Programming, Volume 1, Second Edition, 1998.

[11] "Watcher, NIDS for the masses", Phrack Magazine Vol 8, Issue 53, article 11 of 15.

김 주 영

1998.2 한남대학교 컴퓨터공학과 졸업
2000.2 한남대학교 컴퓨터공학과 석사 졸업
현재 (주)코코넛 연구원
관심분야:침입탐지, 해킹방지, 침입차단
E-mail:jykim@coconut.co.kr

강 창 구



1979 한국항공대학교 항공전자공학과 공학사
1986 충남대학교 전자공학과 공학석사
1993 충남대학교 전자공학과 공학박사
1987.4~2000.3 한국전자통신연구원(ETRI) 국가보안기술연구소 팀장 책임연구원
2000.4~현재 (주)니츠 정보보호기술연구소장
관심분야:암호기술 및 인증 기술, 디지털 서명, 보안 프로토콜, 전산시스템 보안, 네트워크 보안, 키 관리 기술
E-mail:ogkang@nitz.co.kr

소 우 영

1979.2 중앙대학교 전자계산학과 이학사
1981.2 서울대학교, 전자계산학과 이학석사
1991.1 University of Maryland, 전자계산학과 이학박사
1991~현재 한남대학교 컴퓨터전자통신공학부 컴퓨터공학전공 부교수
관심분야:인공지능, 정보보안
E-mail:wsoh@neuro.hannam.ac.kr

이 일 근

1982 경북대학교 전자공학과 공학사
1986 Oregon State Univ. 전자공학 공학석사
1993 Oregon State Univ. 전자공학 공학박사
1990~현재 한남대학교 컴퓨터전자통신공학부 전자공학전공 교수
관심분야:신호처리, 통신공학
E-mail:ikrhee@eve.hannam.ac.kr

이 극

1983 경북대학교 전자공학과(전산모듈) 공학사
1986 서울대학교 컴퓨터공학과 공학석사
1993 서울대학교 컴퓨터공학과 공학박사
1988~현재 한남대학교 컴퓨터전자통신공학부 컴퓨터공학전공 교수
관심분야:보안 시스템, 인공지능, 멀티미디어, 음성인식
E-mail:leegeuk@ce.hannam.ac.kr