SSFNet 기반 사이버 공격 및 탐지를 위한 네트워크 시뮬레이터의 구현

(Implementation of a Network Simulator for Cyber Attacks and Detections based on SSFNet)

심 재 홍 † 정 홍 기 †† 이 철 원 †††
(Jae-Hong Shim) (Hong-Ki Jung) (Cheol-Won Lee)
최 경 희 †††† 박 승 규 †††† 정 기 현 †††††
(Kyung-Hee Choi) (Seung-Kyu Park) (Gi-Hyun Jung)

요 약 가상 공격을 수행하고 이에 따른 네트워크의 행동 변화를 시뮬레이션하기 위하여는 네트워크 구성요소들의 특성을 시뮬레이션 모델에 반영할 수 있어야 하며, 다양한 사이버 공격과 이를 방어하는 시스템들의 특성을 표현할 수 있어야 한다. 본 연구에서는 사이버 공격시 네트워크의 부하가 어떻게 변하는지를 실험하기 위하여, 프로세스 기반 사건 중심 시뮬레이션 시스템인 SSF[9, 10]를 확장 구현하였다. 사이버 공격을 시뮬레이션하기 위해 보안 관련 클래스인 방화벽과 공격용 프로그램을 작성하기 위한 도구들의 모임인 패킷 조작기를 SSF의 구성요소인 SSFNet에 새로이 추가하였다. 이는 보안 체계를 가진 네트워크을 시뮬레이션 가능하게 할 뿐 아니라, 기존 사이버 공격 프로그램을 쉽게 이식하여 시뮬레이션에 적용할 수 있는 장점을 제공한다. 추가된 클래스들의 작동을 검증하기 위하여 가상 네트워크를 구성한 후, 대표적인 서비스-거부 공격인 smurf 공격을 시뮬레이션하고, 이 때의 네트워크의 행동 변화를 관찰하였다. 실험 결과 본 연구에 의하여 개발된 방화벽이나 패킷 조작기가 정상적으로 작동됨을 확인할 수 있었다.

키워드 : 사이버 공격, 공격 탐지, 시뮬레이션, 확장 가능한 시뮬레이션 프레임웍

Abstract In order to simulate cyber attacks and predict network behavior by attacks, we should represent attributes of network components in the simulation model, and should express characteristics of systems that carry out various cyber attacks and defend from these attacks. To simulate how network load may change under the cyber attacks, we extended SSF[9, 10] that is process-based event-oriented simulation system. We added a firewall class and a packet manipulator into the SSFNet that is a component of SSF. The firewall class, which is related to the security, is to simulate cyber attacks, and the packet manipulator is a set of functions to write attack programs for the simulation. The extended SSFNet enables to simulate a network with the security systems and provides advantages that make easy to port already exsiting attack programs and apply them to the simulation evironment. We made a vitual network model to verify operations of the added classes, and simulated a smurf attack that is a representative denial of sevive attack, and observed the network behavior under the smurf attack. The results showed that the firewall class and packet manipulator developed in this paper worked normaly.

Key words : cyber attacks, attack detections, simulation, scalable simulation framework(SSF)

· 본 연구는 조선대학교 및 국가지정연구실 사업에 의해 지원되었습니다.
† 정 회 원 : 조선대학교 인터넷소프트웨어공학부 교수
jhshim@chosun.ac.kr
†† 비 회 원 : 아주대학교 정보및컴퓨터공학부
keeper@madang.ajou.ac.kr
††† 비 회 원 : 국가보안기술연구소 정보보증연구부 연구원
cheolee@etri.re.kr
†††† 종신회원 : 아주대학교 정보및컴퓨터공학부 교수
khchoi@madang.ajou.ac.kr
sparky@madang.ajou.ac.kr
††††† 비 회 원 : 아주대학교 전자공학부 교수
khchung@madang.ajou.ac.kr
논문접수 : 2002년 1월 24일
심사완료 : 2002년 4월 9일

1. 서 론

네트워크 공격과 이에 따른 네트워크 행동 변화에 대한 연구는 공격과 방어를 위한 기반 연구이다[1, 2]. 그러나 현대 네트워크의 방대함, 공격과 방어에 대한 이론적 체계의 부족, 공격과 방어의 복잡성과 다양성, 그리고 공격에 대한 정보 부족 등으로 인하여 아직까지 많은 연구의 진전이 없는 분야이기도 하다. 특히, 네트워크 공격을 탐지하고 대응하기 위하여는 사이버 공격이 시도될 때, 통신량의 변화, 패킷 송신 IP 주소의 분포, 패킷 누락(drop)률의 변화, 혹은 서비스 거부률의 변화 등 네트워크을 구성하는 제반 서브 시스템들에 대한 통계나 시스템 부하 변화에 대한 기초 자료가 필요하다. 특정 공격에 대한 네트워크 행동 변화에 대한 자료를 얻기 위하여, 기존 네트워크에 실제 공격을 시도하고 그 때의 네트워크 행동 변화를 관찰하는 방법이 가장 좋은 방법이다. 그러나 모든 발생 가능한 공격을 시도하여 필요한 자료를 얻는 것은 현실적으로 여러가지 제약사항들이 따른다. 이에 대한 좋은 대안으로 시뮬레이션 기법을 들 수 있으며, 여러 학자들에 의하여 다양한 연구가 진행되고 있다[3].

가상 공격을 수행하고 이에 따른 네트워크의 행동 변화를 시뮬레이션하기 위하여는 다음과 같은 요건들이 만족되어야 한다. 첫째, 시뮬레이션의 대상이 되는 네트워크의 실제 모습을 모델링할 수 있어야 한다. 특히 세계적인 규모의 방대한 인터넷을 모델링할 수 있어야 하며, 모델링된 네트워크상에서 시뮬레이션이 가능하여야 한다. 둘째, 네트워크를 구성하는 서브 시스템들의 배치와 그들의 특성을 네트워크 모델에 반영할 수 있어야 한다. 예를 들면, 운영체제의 통신 프로토콜 스택상에서 일어나는 다양한 패킷의 처리 절차를 표현할 수 있어야 한다. 이는 대부분의 네트워크 공격이 프로토콜 스택에서의 처리를 기반으로 이루어지기 때문이다. 셋째로는 다양한 사이버 공격의 특성에 가깝도록 가상 공격의 시뮬레이션이 가능하여야 한다. 단순한 통계적 기법에 의한 공격용 패킷의 생성보다는 실제로 실행되는 공격 프로그램들이 공격 때에 생성하는 패킷의 모습에 가까운 순서로 패킷을 생성하는 환경 아래에서 가상 공격을 시뮬레이션하는 것이 필요하다. 예를 들면, 포트(port) 스캐닝을 방어하는 연구를 위하여는 이들 스캐닝 도구들이 사용하는 패킷들의 송신 IP들을 실제 시뮬레이션에서 사용할 필요가 있다. 넷째로는 네트워크 공격을 방어하는 시스템들의 특성을 적용할 수 있어야 한다. 예를 들면, 네트워크의 보호를 위하여 가장 널리 사용되는 시스템인 방화벽(firewall)을 네트워크 모델에 반영할 수 있어야 하고, IDS(Intrusion Detection System)가 설치되었을 때의 방어 수준을 연구하기 위해 실제 상황을 시뮬레이션할 수 있는 기반도 필요하다[4-6].

그러나 이러한 성격을 모두 만족하는 시뮬레이션 기반을 확보한다는 것은 대단히 어려운 일이며, 아직까지 대부분의 가상 공격과 이의 시뮬레이션에 대한 연구는 위에서 열거한 기준을 모두 제공하는 환경에서 시뮬레이션한 것이기보다는 한정된 일정 분야에 대해 시뮬레이션을 수행한 연구 결과이다[7, 8]. 이에 본 연구에서는 대규모 인터넷에서의 공격과 이에 따른 네트워크 행동의 시뮬레이션을 위해 기존 SSF(Scalable Simulation Framework)을 확장하여 사이버 공격 및 탐지 시뮬레이션이 가능한 새로운 시뮬레이터 및 실행 환경을 구현하였다. SSF는 프로세스 기반 이산 사건 중심 시뮬레이션 커널(process-based discrete event-oriented simulation kernel)이며, 이를 기반으로 구현된 상위단계의 SSFNet 구성요소와 더불어 100,000개 이상의 노드로 구성된 대규모 네트워크까지도 표현할 수 있다. SSFNet은 라우터(router), 링크(link), 네트워크 인터페이스 카드(NIC) 등 대부분의 네트워크 서브 시스템들을 시뮬레이션하는 객체들이 Java로 구현되어 있으며, 확장성이 뛰어나고 그들의 특성과 행동의 변경이나 확장도 가능하다. 또한 TCP, UDP, ICMP, IP 등 네트워크 프로토콜 스택도 Java로 구현되어 있어, 이들 프로토콜 특성을 이용하는 공격에 대한 시뮬레이션도 가능하다는 장점이 있다. 그외 시뮬레이션용 응용 프로그램 역시 Java로 작성하여 SSFNet와 연동이 가능하며, 다양한 네트워크 행동을 관찰하는 실험을 가능하게 한다[9, 10].

SSFNet의 이러한 장점에도 불구하고 현재 공개되어 있는 SSFNet을 이용하여 네트워크 공격과 이에 따른 네트워크의 행동을 시뮬레이션하기에는 아직 보완되어야 할 사항이 있다. 공격용 프로그램을 Java로 작성하고 이들이 SSFNet에서 구동하기 위하여는 SSFNet에 구현된 프로토콜 스택을 사용할 수 없다. 예를 들면, 패킷 가로채기(capture) 라이브러리에 해당하는 클래스가 없어서 현실적으로 공격용 프로그램을 작성하기 어렵다. 또한 SSFNet에 보안용 시스템을 모델링하는 구성요소를 추가하여 방화벽과 같은 보안시스템이 갖추어진 네트워크를 시뮬레이션할 수 있어야 한다. SSFNet에 의하여 처리되는 프로토콜 스택도 일반적인 운영체제에서 처리되는 TCP 프로토콜 규격을 모두 만족하고 있는 것은 아니어서, 프로토콜 스택의 보안상 결함을 시뮬레이션하기 위해서는 SSFNet의 수정도 필요한 실정이다.

본 논문의 구성은 다음과 같다. 2장에서는 사이버 공격 및 탐지를 위한 시뮬레이터 구조를 기술한다. 3장에서는 가상 공격 및 탐지의 시뮬레이션을 위하여 SSFNet이 어떻게 확장되었는지를 기술하고, 4장에서는 본 연구에서 확장한 시스템의 검증을 시도한다. 끝으로 5장에서 결론을 기술한다.

2. 사이버 공격 및 탐지 시뮬레이터

본 연구에서는 네트워크 공격에 따른 네트워크 행동을 연구하는 기반을 확보하기 위해서 기존 SSF의 구성요소인 SSFNet을 확장 구현하였다. <그림 1>은 SSF와 SSFNet을 기반으로 한 사이버 공격과 이에 대한 네트워크의 행동 시뮬레이션 모델 구조를 보여주고 있다. 흰 바탕의 사각형에 기술된 클래스(class)는 SSF와 SSFNet에 이미 구현되어 공개된 클래스들을 가리키며, 바탕이 어두운 사각형은 본 연구에서 추가적으로 구현한 클래스들을 나타낸다.

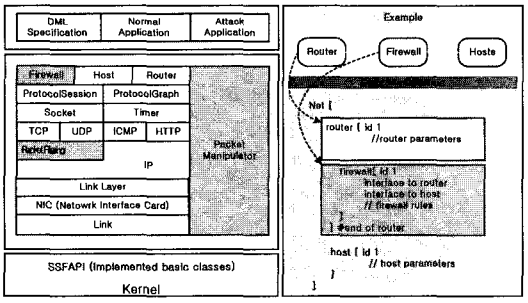

그림 1 SSF와 SSFNet을 기반으로 하는 확장 시뮬레이터 구조도

본 연구에서는 보안 시스템으로서 가장 널리 쓰이고 있는 네트워크 서브 시스템을 시뮬레이션하기 위하여 패킷 필터(packet filter) 기반 방화벽을 추가하였으며, 사이버 공격 프로그램들을 지원하기 위하여 패킷 조작기(*packet manipulator*)를 추가하였다.

이들 두개의 모듈은 SSFNet처럼 자바로 작성되었다. 방화벽 클래스는 SSFNet의 IP 클래스를 확장하여 구현하였으며, IP 계층을 지나가는 패킷들의 송신 및 목적지 IP 주소, 포트번호, 프로토콜(TCP/UDP, IP, ICMP) 종류 및 플래그(flag), 패킷 헤더내의 IP 플래그 등을 바탕으로 패킷을 필터링한다. 필터링 규칙은 LINUX의 ipchains와 동일한 규칙을 사용한다. 필터링 규칙은 SSFNet에서 네트워크의 모델을 기술할 때 사용된 DML (Domain Modeling Language)로 기술하도록 설계하였다. 방화벽 클래스에 관한 자세한 내용은 3장에서 다룬다.

패킷 조작기는 공격용 프로그램들로 하여금 인터넷 패킷을 자유로이 조작하도록 지원하는 인터페이스들을 정의하는 클래스로써 인터넷 패킷의 생성은 물론 패킷 내의 IP 헤더의 제반 정보는 물론 TCP 헤더 및 TCP 데이타를 공격 프로그램에서 자유로이 조작하도록 허용하는 API를 제공하고 있다. 또한 pcap, tcpdump 등과 같은 패킷 가로채기(capturing) 라이브러리처럼 패킷 가로채기 인터페이스를 지원한다. 패킷 가로채기 API는 해당 호스트로 들어오는 모든 패킷이 NIC 객체를 거쳐 IP 클래스를 통과할 때 패킷 조작기가 정의한 큐 속에 복사되어 공격이나 탐지 프로그램으로 전달된다. 패킷 조작기의 자세한 내용은 4장에서 다룬다.

시뮬레이션시 실행되는 공격용 프로그램은 모두 Java로 작성해야 한다. 시뮬레이션을 실행할 때 어떤 공격 프로그램을 어떤 호스트에서 실행할 것인지는 DML을 이용하여 지정하고, 이는 SSFNet에 의하여 구동된다. 따라서 임의의 공격 프로그램을 임의의 호스트에 임의의 수 만큼 실행 가능하게 하여 대규모의 인터넷에서 다양한 공격과 다양한 공격 저지점에서 공격의 효과를 측정하는 것이 가능하다. 공격의 목표인 타겟 시스템의 IP 주소 등 공격 프로그램의 실행시 필요한 제반 매개변수 값들은 모두 DML을 이용하여 정의한다.

3. 방화벽

방화벽은 내부 네트워크이나 내부 네트워크 내의 시스템들을 보호하기 위해 사용되며 가장 널리 이용되는 보안시스템이다. 따라서 사이버 공격과 그에 따른 네트워크의 행동 변화를 시뮬레이션하기 위해서는 가장 기본적인 보안 시스템이 시뮬레이션하고자 하는 네트워크 모델의 구성요소로 표현될 수 있어야 하는 것은 당연하다. 따라서 본 연구팀은 SSFNet을 확장하여 보안시스템인 방화벽을 모델링 할 수 있는 구성요소를 추가하였다. 본 장에서는 SSFNet에 추가된 방화벽 구성요소에 대해 논의하고, 이를 표현하는 방법에 대해 기술한다.

3.1 방화벽 모델링 구성요소

방화벽 구성요소는 패킷 필터링(packet filtering-based) 기반 방화벽 보안 시스템을 모델링한 클래스이다. 패킷 필터링 방화벽은 방화벽 시스템을 통과하는 패킷들을 허용할 것인지 혹은 버릴 것이지를 알려주는 규칙들의 집합(rule set)을 정의한다. 규칙 집합의 표현 방법은 방화벽 시스템에 따라 다소 차이는 있으나 대부분 규칙 번호, 필터링 형태(incoming, outgoing, 혹은 forwarding 패킷 여부), 송신자 및 수신자의 IP 주소와 포트 번호, 통

신 프로토콜, 그리고 규칙과 일치하는 패킷의 처리 방법(본 연구에서는 패킷의 폐기를 의미하는 거부(deny), 패킷의 통과를 허용하는 허용(accept)) 등으로 정의된다.

본 연구에서는 방화벽을 하나의 독립된 구성요소가 아닌 IP 계층을 확장하는 방법으로 구현하였다. 따라서 기존 SSFNet에 존재하는 IP 클래스를 상속받아 방화벽 구성요소로 SecureIP 클래스를 추가하였다. 패킷의 기본적인(default) 처리방법은 거부 방법을 사용하며, 따라서 통과시켜야 하는 패킷들은 규칙 집합 내에 반드시 포함시켜야 한다. 이러한 점을 고려한 규칙 집합은 앞서 기술한 DML 내에 방화벽을 위한 규칙으로 별도로 기술되며, 이 경우 네트워크 시뮬레이션 내에 방화벽이 존재하는 좀 더 현실에 가까운 시뮬레이션 환경을 구축할 수 있다.

3.2 방화벽 클래스 구조

<그림 2>는 본 연구에서 제안한 SecureIP 클래스의 필터링 규칙 적용의 흐름을 보여준다. 방화벽을 모델링하는 이 클래스는 IP 계층으로 전달되는 모든 패킷의 헤더를 분석하여 자신의 컴퓨터로 들어오는지(IN), 자신의 컴퓨터에서 나가는지(OUT), 또는 다른 네트워크로 전달(FWD)해야 하는지를 결정하며, 각각에 관련되는 필터링 규칙 집합을 면밀히 검사한다. 필터링 규칙을 검토한 결과가 '허용(permit)'이라면 패킷은 정상적으로 다음 단계에 보내지게 되며, 그렇지 않은 경우에는 패킷을 폐기한다[11].

SecureIP 클래스를 이용하여 방화벽을 모델링하기 위하여는 기존의 IP 클래스를 SecureIP 클래스로 대체하여 사용해야 한다. 따라서 기존의 IP 클래스나 또는 SecureIP 클래스를 사용하는 것은 상위계층의 응용보다 하위 계층에서 정의되는 것이므로, 시뮬레이션용 응용 프로그램 개발자들은 방화벽을 고려하지 않고 투명하게 프로그램을 작성할 수 있다. 단, 방화벽을 시뮬레이션하고자 할 경우, DML 내의 라우터를 기술하는 부분에서 ProtocolSession 내의 IP 클래스로 기존 IP 클래스 대신 SecureIP 클래스로 지정하여야 하고, 아울러 필터링 규칙도 함께 정의한다.

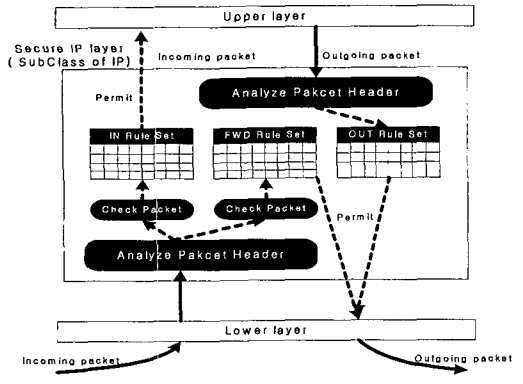

그림 2 방화벽을 위한 SecureIP 클래스의 필터링 규칙 적용의 흐름

3.3 방화벽의 DML 표현 예제

<그림 3>은 방화벽을 정의하는 DML의 예를 보여주고 있다. 그림에서 DML이 정의하는 라우터 모델은 두 개의 인터페이스를 가지고 있다. 하나는 10Mbps의 속도를 가지는 내부 네트워크와 연결된 인터페이스고, 다른 하나는 100Mbps의 속도를 가지는 외부 네트워크와 연결된 인터페이스이다. 내부 네트워크의 주소가 210.1.30.1~210.1.30.254 까지를 사용한다고 가정하고 있으며, 두 개의 필터링 규칙을 정의하였다. 첫 번째 규칙은 단순하게 내부 네트워크 내에서 TCP용 ftp 서비스를 제한시키는 규칙으로서 패킷의 목적지 주소가 자신의 내부 네트워크이고, 프로토콜은 TCP이며, 포트 번호가 21(ftp 서비스)인 경우 모든 패킷을 폐기함을 의미한다. 두 번째 규칙은 DOS 공격에서 많이 사용하는 ICMP 패킷을 제한하기 위한 규칙이다. 공격자가 소스 주소를 내부 네트워크으로 바꾼 ICMP 패킷을 내부 네트워크로 보내면, 라우터에서는 목적지 주소만을 가지고 판단하기 때문에 내부에서 내부로 전달된 패킷으로 간주한다. 방화벽에서 내부 네트워크에서 내부 네트워크로

```
router [ id 1
    interface [ id 0 _extends .dictionary.10Mb]
    interface [ id 1 _extends .dictionary.100Mb]
    graph [
        # Protocol Session Definition.
        # 방화벽을 사용하려면, 반드시 ip 레이어를 책임지고 있는 클래스를
        # SSF.OS.SecureIP 클래스로 설정해야 한다.
        ProtocolSession [name ip use SSF.OS.SecureIP
            Firewall[
                Rule[
                    # 자신의 내부 네트워크에서 TCP용 ftp 서비스를 금지
                    Type        FWD
                    id          1
                    SrcAddr     *
                    SrcPort     *
                    DestAddr    210.1.30.*
                    DestPort    21
                    UseProtocol TCP
                    Direction   *
                    Action      Deny
                ]
                Rule[
                    # 외부로부터 온 패킷이지만, 소스 주소가 자신의 내부에
                      존재하는 호스트 주소로 되어 있는 경우 금지
                    Type        FWD
                    id          2
                    SrcAddr     210.1.30.*
                    SrcPort     *
                    DestAddr    210.1.30.*
                    DestPort    *
                    UseProtocol *
                    Direction   Outbound
                    Action Deny
                ]
            ] #end of Firewall
        ] #end of ProtocolSession ip
        ProtocolSession [name icmp use SSF.OS.OSPF]
    ] #end of graph
] #end of router
```

그림 3 방화벽을 모델링하는 DML 표현 예제

의 데이타 전송을 허용하는 맹점을 노린 공격 패턴이라고 할 수 있다. 이러한 경우를 방지하기 위해서는 방화벽에서 두 주소 부분이 내부로 표시되어 있음에도 불구하고, 패킷이 전달된 경로가 외부 인터페이스로부터 전달이 되었다면 해당 패킷을 폐기한다. 두 번째 규칙은 이러한 종류의 공격을 막기 위한 규칙을 설정하는 예를 보여주고 있다.

규칙들의 개수에는 제한이 없으며, 다음과 같은 점을 고려하여 사용자가 DML 상에 기술하여야 한다. 첫째, 패킷이 어느 규칙과도 일치하지 않는다면 SecureIP 클래스에서는 패킷을 폐기하는 것을 원칙으로 한다. 둘째, 패킷은 사용자가 정의한 규칙의 순서에 따라 비교되며 일치하는 규칙이 존재할 경우, 그 즉시 해당 규칙에서 설정한 조치(action)를 취한다. 따라서 이러한 규칙을 정의하는 순서가 중요하다.

4. 패킷 조작기(Packet Manipulator: PM)

패킷 조작이란 패킷의 생성, 원하는 패킷 필드의 설정(set), 수정(update), 정보추출(get) 등을 의미한다. SSFNet은 기존 소켓 API를 이용하여 TCP, UDP 패킷을 전송하거나 수신하는 정도의 기본 API만 제공할 뿐, 패킷의 헤더를 직접 조작할 수 있는 API와 기능은 제공하고 있지 않다. 이에 본 연구에서는 공격자들이 즐겨 사용하는 패킷 조작을 가능하게 하고, 이를 통한 공격을 시뮬레이션이 가능하도록 SSFNet 내에 별도의 패킷 조작기 구성요소를 개발하여 추가하였다. 본 장에서는 SSFNet에 추가된 패킷 조작기 구성요소에 대해 논의하고, 이를 이용한 예제 시뮬레이션 프로그램에 대해 기술한다.

4.1 패킷 조작기 기능

한 ICMP 패킷을 생성하고, 이를 조작할 수 있는 기능을 제공한다. 이는 IP 헤더, ICMP 메시지 등으로 구성된 완전한 하나의 패킷을 생성하고, 이를 조작하는 것이다. 즉, 앞서 언급된 IP 조작에서 제공하는 기능을 이용하여 기본적인 IP 패킷이 제작되면, 이를 이용하여 ICMP 패킷을 손쉽게 제작할 수 있다. 이를 위해선 사용자는 우선 IP 패킷을 생성하고, 이를 ICMP 패킷 제작시 활용하도록 한다. 또한 ICMP 메시지 구성 요소인 패킷 타입, 코드, 체크 합계, 기타 타입과 코드에 의존하는 데이트 등을 설정하고 수정할 수 있다. ICMP 패킷은 특히 smurf 공격에서 자주 사용된다.

이상의 패킷 조작 기능에 의해 제작된 패킷은 전용 송신 API를 통해 전송해야 한다. 이는 위에서 제작된 IP, UDP, TCP, ICMP 패킷은 SSFNet가 제공하는 기존 소켓 API를 통해서는 전송할 수 없으며, 별도의 경로를 따라 송신되어야 하기 때문이다. 제공된 전송 송신 API는 경우에 따라 전송된 패킷에 대한 응답을 기다리거나, 또는 바로 리턴할 수 있는 옵션도 지원한다.

끝으로 공격의 필수적 요건으로 공격에 대한 응답 패킷을 수신하는 기능이 필요하다. 이를 위하여 PM은 패킷 받기와 패킷 가로채기(packet capturing)을 기능을 지원한다. 패킷 보내기에 의해 보내진 패킷은 경우에 따라 대상 호스트로부터 응답이 올 수 있으며, 이를 받아 다음 행위를 결정할 수 있어야 한다. 따라서 시뮬레이션 역시 이러한 패킷을 받을 수 있는 메커니즘을 제공하여야 한다. 기존의 SSFNet 소켓 API는 해당 포트로 전송된 데이타만 받을 수 있을 뿐, 단순한 IP 패킷을 응용에서 임의로 받을 수 있는 API는 제공하지 않고 있다. 전통적으로 UNIX 시스템에서는 네트워크상의 모든 패킷을 잡아 낼 수 있는 패킷 필터를 제공하며, 이러한 패킷 필터를 이용하여 사용자는 원하는 패킷을 받아 낼 수 있었다.

그러나 본 연구에서는 네트워크상의 모든 패킷을 잡아 내는 패킷 필터링 기능은 라우터, 방화벽 등에서 활용하는 것이 적절하고, 공격을 위해선 패킷 필터보다는 자신의 호스트로 전송된 패킷만을 잡아 낼 수 있는 패킷 가로채기 기능이 보다 효율적이다. 따라서 네트워크상의 모든 패킷이 아닌 자신의 호스트에 도착한 패킷만 잡아내는 필터링 기능을 지원하도록 하였다.

4.2 패킷 조작기의 구현

<그림 4>는 PM에서 패킷을 생성하는 클래스들간의 관계도를 보여주고 있다. 새로운 패킷을 만들기 위해서는 IP 헤더를 생성한 후, 각각의 프로토콜 클래스에게 파라미터로 전달한다. 각각의 프로토콜 생성 클래스에서

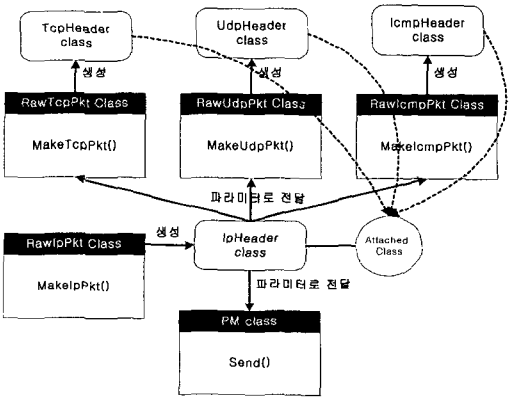

그림 4 PM 클래스와 관련된 클래스의 계층도 및 흐름도

는 자신의 헤더를 생성하고, 전달받은 IP 헤더의 페이로드(payload)로 삽입되어, 하나의 IP 패킷을 생성하게 된다. 이런 과정을 통해 생성된 IP 패킷은 PM 클래스를 통해 원거리 호스트로 보내진다.

RawIpPkt 클래스는 SSFNet에서 제공하는 IP 헤더 클래스처럼 실제 네트워크 상에서 사용되는 IP 헤더를 간략화 하여 제공한다. RawIpPkt 클래스에서는 일반적인 TCP/IP 혹은 UDP/IP 에서 사용되는 제반 값들을 프로그래머가 직접 설정할 수 있도록 제공한다. RawIp Pkt에서 IP 헤더를 생성하는 메소드는 다음과 같이 호출되어 진다.

```
RawIpPkt r_ip = new RawIpPkt();
// 파라미터 값들을 설정한다. protocol_no, src_ip, dest_ip,
   total_length, tos, ttl
r_ip.MakeIpPkt();
```

RawIcmpClass는 SSFNet에서는 IP와 마찬가지로, ICMP 패킷도 간략화하여 구현되어 있다. RawIcmpPkt 클래스에서는 ICMP 파라미터들을 사용자가 정의한 후에, ICMP 패킷을 생성하는 기능을 제공하며, 이와 같이 생성된 헤더는 파라미터로 전달받은 IP 헤더 뒤에 연결하여 완전한 IP 패킷을 만들어 낸다. RawIcmpPkt 에서 ICMP 패킷을 생성하는 방법은 다음과 같다.

```
RawIcmpPkt r_icmp = new RawIcmpPkt();
// 파라미터 값들을 설정한다. icmp_type, icmp_code, id, seq,
   datalen, originTimestamp,
// receiveTimestamp, transmitTimestamp
r_icmp.MakeIcmpPkt(r_ip /* 생성한 IP Header */);
```

RawTcpPkt 클래스에서는 TCP 헤더를 생성하는 기능을 제공하며, 생성된 TCP 헤더를 파라미터로 전달받은 IP 헤더 뒤이 연결하여 완전한 IP 패킷을 만들어 낸다. RawTcpPkt에서 TCP 헤더를 생성하는 방법은 다음과 같다.

```
RawTcpPkt r_tcp = new RawTcpPkt();
// 파라미터 값들을 설정한다. smsg, srcPort, destPort, seqNo,
   ackNo, advertiseWnd, flags
r_tcp.MakeTcpPkt(r_ip /* 생성한 IP Header */);
```

RawUdpPkt 클래스에서는 위의 파라미터들을 정의한 후에, UDP 헤더를 생성하는 기능을 제공하며, 생성된 UDP 헤더를 파라미터로 전달받은 IP 헤더 뒤에 연결하여 완전한 IP 패킷을 만들어 낸다. RawUdpPkt에서 UDP 패킷을 생성하는 방법은 다음과 같다.

```
RawUdpPkt r_ucp = new RawUdpPkt();
// 파라미터 값들을 설정한다. dmsg, srcPort, destPort, length
r_udp.MakeUdpPkt(r_ip /* 생성한 IP Header */);
```

위에서 제시한 방법으로 만들어낸 IP 패킷은 실제로는 네트워크상에 존재하는 원거리 호스트에게 보내야 한다. 그러나 SSFNet에서의 IP 클래스는 실제의 IP와 같이 IP 레이어에서 새로운 IP 패킷을 만들어서 보내는 독립적인 기능을 가지고 있지 않다. SSFNet에서 IP 헤더를 생성하는 곳은 상위 프로토콜 (TCP, UDP, ICMP 등)에서 생성하며, IP에서는 패킷을 보내고 받는 부분만을 책임지고 있다. 따라서 본 연구에서도 공격자가 프로토콜 스택을 따르지 않고 임의로 생성한 패킷을 실제로 NIC(Network Interface Card)를 통해 직접 보내지 않고, SSFNet에서 제공하는 IP 레이어를 통하여 보내는 방법을 채택하여 구현하였다. PM 클래스를 통하여 패킷을 보내는 방법은 다음과 같다.

```
PM pm = new PM(this /* 호출한 인스턴스의 정보를 알려줌 */);
pm.Send( r_ip /* 완성된 ip 패킷 */,
         this /* 보낸 패킷을 되 돌려 받았을 때, PM에 의해서 호
                 출될 인스턴스 */ )
```

공격자는 자신이 생성한 패킷이 정상적인 프로토콜 스택을 따르지 않고 보냈듯이, 자신의 컴퓨터로 들어오는 패킷을 정상적인 프로토콜 스택을 거쳐 해석된 상태의 데이타가 아니라, 해석되지 않은 원래의 데이타를 분석할 필요가 있다. 일반적인 운영체제에서도 이러한 기능을 커널 레벨에서 지원을 해주고 있으며, 이를 promiscuous 모드라고 한다. 본 연구에서는 SSFNet에서 이와 같은 기능을 지원할 수 있도록 구현하였다.

<그림 5>는 정상 모드와 패킷 가로채기 모드일 때의 패킷 흐름을 보여준다. 실선은 일반적인 패킷의 흐름을

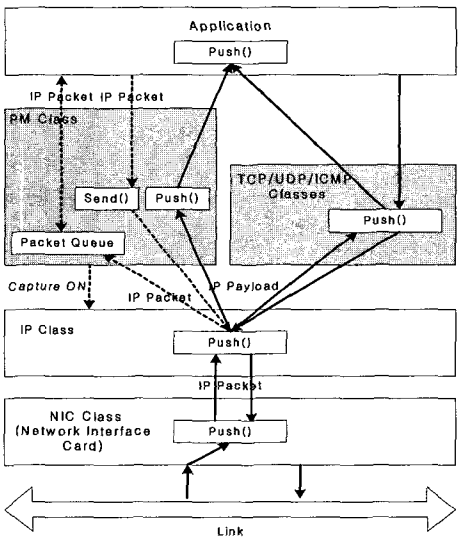

그림 5 정상 모드와 패킷 가로채기 모드일 때의 패킷 흐름

나타내고 있으며, 점선은 패킷 가로채기 모드일때의 패킷 흐름을 보여준다. 그림에서 보듯이 PM을 사용할 때도, 가로채기 모드가 OFF라도 자신이 보낸 패킷에 대한 응답에 해당하는 패킷은 정상적으로 push() 메소드를 통해서 어플리케이션 프로그램까지 전달된다. 따라서 이 경우 IP 계층에서의 패킷 흐름은 PM 계층이나 또는 정상적인 프로토콜 계층 중 한 방향으로 흘러간다. 그러나 가로채기 모드가 ON인 경우에는 PM의 패킷 큐와 PM의 push() 그리고 PM의 패킷 큐와 정상적인 프로토콜 계층 양 방향으로 흘러가게 된다.

4.3 패킷 조작기를 이용한 예제 프로그램

<그림 6>은 PM이 제공하는 API를 이용한 smuf 공격 시뮬레이션 프로그램 예제이다. smurf 공격은 ICMP 메아리 요청(echo request) 패킷을 만들어, 이를 IP network-directed broadcast 주소의 중간 네트워크로 브로드캐스팅하며, 이 패킷을 받은 중간 네트워크의 모든 시스템들은 이 패킷의 소스 주소(위장된 공격 대상의 시스템 주소임)로 메아리(echo) 응답용 ICMP 패킷을 보내어, 공격 대상이 되는 호스트에 과도한 CPU 또는 네트워크 부하가 걸리게 함으로써 서비스를 못하게 하는 공격이다. 이 예제는 공격에 이용하는 브로드캐스팅용 중간 네트워크를 하나만 이용하지만, 경우에 따라 파상적인 공격을 위해 여러 개의 중간 네트워크를 활용할 수도 있다.

이 프로그램은 우선 DML 파일로부터 공격할 대상 호스트의 주소와 브로드캐스팅시킬 중간 네트워크의 주소를 얻는다. 또한 전송할 패킷의 수와 패킷간의 송출 시간 간격, 패킷 크기 등을 DML 파일로부터 읽어 들인다. 브로드캐스트 주소는 network-directed 브로드캐스트 주소로서 공격을 가하는 시스템이 소속된 네트워크에 브로드캐스팅 시키는 것이 아니라, 제3의 네트워크에 브로드캐스팅 시킨다. 프로그램은 먼저 IP 헤더를 만들고, 이를 이용하여 IP 헤더, ICMP 헤더, ICMP 데이타로 구성된 ICMP 메시지를 만든 다음, 이를 주기적으로 bcastaddr로 num번 전송한다.

5. 구현 시스템의 검증

본 장에서는 본 연구에서 구현한 방화벽과 패킷 조작기 구성요소들을 동작 상태를 검증하기 위하여, 우선 인터넷 가상 통신망을 모델링하고, 네트워크 공격이나 방화벽과 같은 실 세계 네트워크 상에서 나타날 수 있는 특성들을 표현하여 이를 시뮬레이션함으로써, 구현한 사

```
/* define packet manipulatior instance */
PM pm = new PM(this);

/* get following items from DML file
target = get target host IP address to hit
bcastaddr = get network-directed broadcast address(x.x.x.255)
num = get number of packets to send (0 = flood)
pktdelay = get packet delay(wait) between each packet (in ms)
pktsize = get packet size ;
/* Using Timer class, we will send the packet periodically */
sequence = 1;
(new Timer(inGraph(),Net.seconds(pktdelay)){
   public void callback() {
      try {
         /* make RawIpPkt instance */
         RawIpPkt r_ip = new RawIpPkt();
         /* Here, you must set RawIpPkt class fields */
         /* total length, src ip, dest ip, protocol no,
            tos, ttl */
         IpHeader ip_h = r_ip.MakeIpPkt();
         /* make RawIcmpPkt instance */
         RawIcmpPkt r_icmp = new RawIcmpPkt;
         /* Here, you must set RawIcmpPkt class fields */
         r_icmp.icmpType = ICMP.ICMP_ECHO_REQUEST;
         r_icmp.id = 1; r_icmp.seq = sequence++; r_icmp.datalen=pkt_size;

         r_icmp.MakeIcmpPkt(ip_h);
         /* send this packet to remote hosts */
         pm.send(ip_h, this);
         num --;
         if ( num > 0 )
                  /* setting Timer again */
                  set();
      } catch (ProtocolException pex) {
         pex.printStackTrace();
      }
   }
 )).set();

} /* end of methods */

public boolean push(ProtocolMessage msg, ProtocolSession from)
{
  // you can implement this method optionally.
}
```

그림 6 단일 네트워크를 이용한 smurf 공격

이버 공격 및 탐지 시뮬레이터 기능의 타당성을 보이고자 한다.

본 장에서의 시뮬레이션은 구현된 시뮬레이터의 정상적인 동작 여부를 판단하기 위한 실험이며, 네트워크의 부하 및 행동 변경을 분석하기 위한 시뮬레이션은 아니다. 향후 본 연구에서 구현된 시뮬레이터를 기반으로하는 다양한 네트워크 공격과 방어 등에 관한 시뮬레이션 결과와 이에 대한 상세한 분석은 별도의 연구 논문을 통해 발표할 예정이다.

5.1 통신망 구조

시뮬레이션을 위한 네트워크으로 본 연구에서는 SSF.NET 개발자들이 인터넷을 모델링하기 위해 만들었던 네트워크 모델을 사용하였다[7, 8]. 참고문헌[8]에서는 100,000개의 Host로 구성된 거대한 네트워크를 모델링하였지만, 본 논문에서의 시뮬레이션 실험에서는 시뮬레이터의 작동을 검증하는 것이 목적이므로 13,000여개의 클라이언트와 40개의 대형 서버, 270개의 라우터(OSPF)들로 네트워크를 구성하였다.

<그림 7>은 1,300개의 호스트로 구성된 하나의 자치 네트워크를 나타내며 이러한 네트워크의 복사본 10개를 BGP라우터로 연결하여 전체 네트워크를 구성하였다. 호스트와 라우터간의 대역폭은 100Mbps로 라우터와 라우터간은 1Gbps이며 BGP 라우터간은 10Gbps로 설정하였으며, 링크 지연은 구간에 따라 0.001초에서 0.03초까지 설정하였다. 각 호스트는 기본적으로 HTTP 및 TCP/IP 프로토콜을 사용하여 전체 서버에 대하여 서비스를 요청하도록 구성하였다.

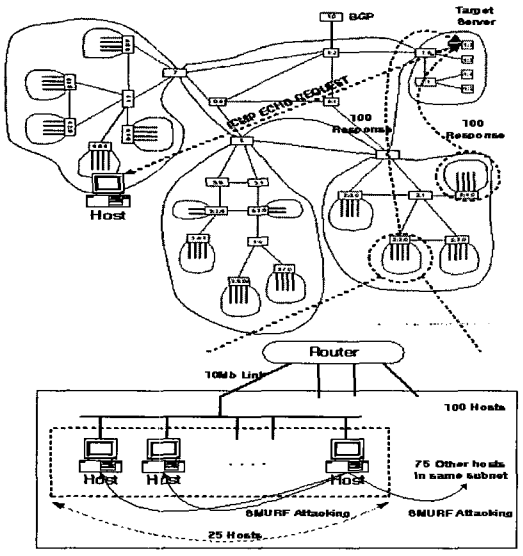

그림 7 자치 네트워크 구성도

그 밖에 본 연구에서 사용된 모델의 프로토콜 파라미터 그리고 프로토콜들의 구성은 참고문헌 [8]과 동일하게 설정하였다. 본 연구에서는 패킷 전송상에는 에러가 발생하지 않으며, 시간 지연으로 인한 타임아웃에 의한 패킷 손실은 없다고 가정한다.

5.2 검증 방법

구현된 시뮬레이터 동작의 검증 환경은 다음과 같다. 네트워크 상에 존재하는 클라이언트들은 일반적으로는 HTTP 프로토콜을 이용하여 서버로부터 웹 서비스를 받는다고 가정한다. SSFNet에서는 특정 호스트(서버)의 내부 부하를 측정할 수 있는 방법을 제공하지 않기 때문에, 본 연구에서는 네트워크 부하를 기반으로 서버의 응답시간 지연을 측정하였다. 네트워크 공격 유형으로 대표적인 DDOS 공격인 smurf 공격을 사용하였다. 브로드캐스트 주소의 중간 네트워크 내의 모든 호스트들은 smurf 공격에 의해 전송된 ICMP 메아리 요청 패킷에 반응하여, 하나의 공격대상이 되는 서버(위장된 소스 주소)로 응답 메시지를 보내게 된다. 그 외의 네트워크 상에 존재하는 다른 호스트들은 HTTP를 이용한 웹 서비스를 요청한다. 클라이언트 중 한 시스템은 smurf 공격 목표인 서버에게 정상적인 ping 패킷을 보내서 이의 응답시간의 변화를 측정한다.

본 시뮬레이션에서는 이와 같은 방법으로 smurf 공격 패킷을 받는 중간 네트워크의 수를 변화시켜 가면서 ping 패킷에 대한 서버의 응답시간의 변화를 측정하였다. 또한, smurf 공격을 받는 중간 네트워크의 수를 고정하고, smurf 공격 패킷의 크기를 변화시켜 가면서 ping 패킷에 대한 서버의 응답시간의 변화를 측정했다.

Smurf 공격을 위해 사용된 클라이언트의 DML코드는 <그림 8>과 같다. DML 코드에서 사용하는 smurf 공격을 위한 Java 클래스 파일은 기존의 Ping Client.java를 수정하여 브로드캐스팅과 유사하게 패킷을 중간 네트워크의 호스트로 보내는 기능을 한다.

```
Client10Mb_attack [
   interface [id 0 bitrate 10000000 latency 0.001]
   graph [
      ProtocolSession [name ip use SSF.OS.IP]
      ProtocolSession [name icmp use SSF.OS.ICMP]
                        ProtocolSession [name pingerBroad use
                    pingClientBroad count 500 interval 0.1
                    len 1024 ]
   ] # End of graph
] # End of client
```

그림 8 Smurf 공격을 위해 사용된 클라이언트의 DML 코드

5.3 검증 결과 및 분석

본 연구에서는 smurf 공격시 브로드캐스팅하는 중간 네트워크의 수를 0개에서부터 24개까지 늘려가면서 ping 패킷에 대한 서버의 응답시간 변화를 측정해 보았다. 공격시 전송하는 패킷의 크기는 1024 바이트이며, 측정을 위한 패킷의 크기는 48 바이트의 크기이다. 공격 및 측정을 하는 호스트는 0.1초 단위로 패킷을 생성하여 총 500개의 패킷을 보낸다. 하나의 중간 네트워크에는 100개의 호스트가 존재하므로 0.1초마다 100개씩 패킷을 생성하여 보내는 형태이다. 또한, 이러한 공격을 위하여, 본 연구에서 확장 구현한 PM 클래스를 사용하여 패킷 조작을 하였으며, 이를 통해 smurf 공격 코드를 생성하였다.

먼저 DDOS 공격을 하는 호스트가 없을 경우, 한 호스트에서 일반적인 ICMP 메아리 요청 패킷을 보내어 응답을 받는 시간은 아무런 부하가 없기 때문에 비교를 위한 기본 값이 된다. 시뮬레이션 수행 후 이 응답시간의 평균은 0.794004 이었다. 브로드캐스팅하는 중간 네트워크의 수를 증가시켰을 때 변화한 응답시간은 <그림 9>와 같다.

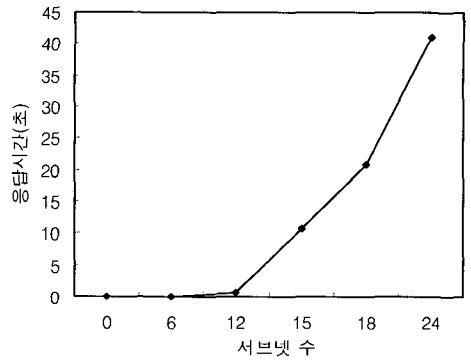

그림 9 Smurf 공격시 중간 네트워크의 수에 따른 ping 패킷의 평균 응답시간 변화

<그림 9>와 같이 하나의 중간 네트워크에 브로드캐스트를 보냈을때의 평균 응답시간과 11개의 중간 네트워크에 브로드캐스트를 보냈을때와는 거의 차이가 없이 일정한 값을 유지하고 있다. 이는 네트워크의 대역폭이 이 정도의 데이타 흐름은 충분히 수용할 수 있음을 알 수 있다. 11개 네트워크에 브로드캐스트를 하였을 때 약간씩의 증가가 있다가 12개가 되었을 때 급격한 증가를 보인다. 이는 이 시점부터 네트워크의 부하가 발생하고 통신망 자체의 데이타 수용량이 한계에 다다랐음을 알 수 있다. 이러한 실험 결과는 사용된 통신망들의 대여폭과 중간 네트워크에 존재하는 호스트의 개수 그리고 공격 대상 호스트의 부하 상태에 따라 다른 결과를 나타낼 수 있다. 그러나 이러한 실험을 통해 공격에 사용되는 각 중간 네트워크 내의 호스트 개수를 파악하고 있을 때, 몇 개의 중간 네트워크를 활용해서 공격을 해야만 공격 대상의 호스트 또는 네트워크가 마비 또는 서비스를 하지 못할 정도가 될 수 있는가를 예측할 수 있게 한다. 역으로 방어하는 측에서는 몇 개의 다른 중간 네트워크를 활용하여 자신의 호스트 또는 지역 네트워크를 공격하는지를 판단할 수 있게 한다.

<그림 10>은 smurf 공격을 받는 중간 네트워크의 수를 고정하고 공격하는 패킷의 크기를 변화시켰을 때, 하나의 호스트에서 보낸 *ping* 패킷의 응답시간의 변화를 측정한 결과를 보여준다. 이 때, 공격하는 중간 네트워크는 12개(<그림 9>에서 네트워크의 과부하가 걸린 시점)로 설정하였다. 그 외의 실험 환경은 앞의 경우와 동일하며 패킷 크기를 기본 ICMP 메아리 요청 패킷 경우인 24 바이트부터 1KB, 1.5KB, 2KB, 5KB, 10KB로 점차 늘려보면서 12개의 중간 네트워크에서 이 크기의 패킷을 동시에 서버로 전송 하였다. 각각의 응답시간의 측정은 하나의 호스트에서 400번째부터 500번째 보낸 ping 패킷의 응답시간 결과의 평균을 사용하였다. <그림 10>에서 패킷의 크기가 커질수록 네트워크상의 부하가 커지므로, 응답시간이 점차 증가되다가 어느 시점부터는 증가 부분이 나타나지 않는 것을 볼 수 있다. 이러한 현상은 각 중간 네트워크에서 보내는 패킷들의 경로가 수렴되는 링크구간의 사용률이 100%로 되는 순간부터 나타나게 되는데, 이 시점에서 병목구간에 존재하는 라우터가 처리하는 패킷의 양이 같아 지기 때문이다. 그러나 공격하는 패킷의 크기가 그 순간부터 의미가 없어

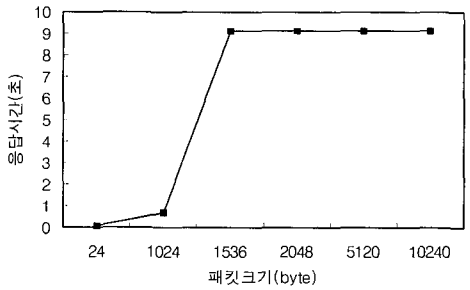

그림 10 Smurf 공격시 ping 패킷 크기에 따른 평균 응답시간

지는 것은 아니다. 패킷의 크기가 커질수록 네트워크의 사용률이 최대값으로 유지되는 시간은 길어지게 되어 서버의 지연시간 및 ping 패킷의 최대 응답시간도 점점 증가된다. 이와 같은 시뮬레이터의 결과를 분석한 결과, 실제 네트워크에서의 현상을 상당히 유사하게 나타낸 것으로 판단되어, 본 구현 시뮬레이터가 제대로 동작하고 있음을 확인할 수 있다.

6. 결 론

본 연구에서는 사이버 공격시 네트워크의 트래픽이 어떻게 변하는지를 실험하기 위한 시뮬레이터를 개발하였다. 이를 위하여 프로세스 기반 사건 중심 시뮬레이션 시스템인 SSFNet를 확장 구현하였다. SSF와 SSFNet의 장점은 살리면서, 사이버 공격을 시뮬레이션 하기에 부족한 보안 관련 클래스인 방화벽과 공격용 프로그램을 작성하기 위한 도구들의 모임인 패킷 조작기를 SSFNet에 추가하였다. 모든 방화벽과 패킷 조작기 구성요소들은 Java로 작성되었으며, 시뮬레이션을 위하여 사용되는 가상 공격 프로그램도 Java로 작성하도록 하였다. 이는 다양한 보안 체계를 가진 네트워크의 실험이 가능할 뿐 아니라, 어떤 사이버 공격 프로그램이라도 쉽게 이식하여 시뮬레이션에 적용할 수 있다는 장점을 제공한다.

추가된 클래스들의 작동을 검증하기 위하여 가상 네트워크를 구성한 후, DDOS(distributed denial of services) 공격으로 널리 알려진 smurf 공격을 시뮬레이션하고, 이 때의 네트워크의 행동 변화를 관찰하였다. 실험 결과 본 연구에 의하여 개발된 방화벽이나 패킷 조작기가 정상적으로 작동됨을 확인할 수 있었다.

현재 본 연구팀은 구현된 사이버 공격 및 탐지 시뮬레이터를 이용하여, 다양한 네트워크 공격용 가상 공격 프로그램을 개발하고, 또한 라우터와 방화벽의 다양한 방어 규칙 집합의 모델을 정의하여, 실제 상황과 유사한 환경에서의 시뮬레이션 실험을 실시하고 있다. 이러한 사이버 공격과 방어, 탐지 시뮬레이션의 결과와 이의 심층 분석 등은 별도의 논문으로 발표할 계획이다.

참 고 문 헌

[1] G. Vigna, S. Eckmann, and R. Kemmerer, "The STAT Tool Suite," In Proceedings of DISCEX 2000, Hilton Head, South Carolina, IEEE Computer Society Press, 2000.

[2] G. Vigna, R.A. Kemmerer, "NetSTAT: A Network-based Intrusion Detection System," Journal of Computer Security, Vol. 7, No. 1, pp. 37-71, 1999.

[3] Secure Networks, Custom Attack Simulation Language(CASL), Jan. 1998.

[4] Brett, "Building Bastion Routers Using Cisco IOS," Phrack Magazine, Vol. 9, 1999.

[5] R. Durst, T. Champion, B. Witten, E. Miller, and L. Spagnuolo, "Testing and Evaluating Computer Intrusion Detection Systems," CACM, Vol. 42, No. 7, pp. 53-61, 1999.

[6] K. Ilgun, "USTAT: A Real-time Intrusion Detection System for UNIX," In Proceedings of the IEEE Symposium on Research on Security and Privacy, Oakland, CA, May 1993.

[7] James H. Cowie, David M. Nicol, and Andy T. Ogielski. "Modeling the Global Internet," Computing in Science & Engineering, pp. 42-50, Jan.~Feb. 1999.

[8] James H. Cowie, David M.Nicol and Andy T. Ogielski. "Modeling 100,000 Nodes and Beyond: Self-Validating Design," DARPA/NIST Workshop on Validation of Large Scale Network Simulation Models, May 25-26, 1999.

[9] James H. Cowie, "Salable Simulation Framework API Reference Manual," Version 1.0, Documentation Draft - Revision, March 1999, http://www.ssfnet.com

[10] "SSF Simulator Implementation," http://www.ssfnet.com/ssfImplementations.html

[11] Brent D. Chapman and Elizabeth D. Zwicky, Building Internet Firewalls, O'Reilly & Association Inc., Sept. 1995.

심 재 홍

1987년 서울대학교 전산과학과 졸업(학사). 1989년 아주대학교 컴퓨터공학과 졸업(석사). 2001년 아주대학교 컴퓨터공학과 졸업(박사). 1989년 ~ 1994년 서울시스템(주) 공학연구소. 2001년 ~ 2001년 9월 아주대학교 정보통신전문대학원 BK21 전임연구원. 2001년 10월 ~ 현재 조선대학교 인터넷소프트웨어공학부 전임강사. 관심분야는 운영 체제, 분산시스템, 실시간 및 멀티미디어시스템

정 홍 기

1996년 아주대학교 정보 및 컴퓨터 공학과 졸업. 1999년 아주대학교 정보 및 컴퓨터 공학 졸업(공학 석사). 1999년 ~ 현재 아주대학교 정보통신 전문대학 박사 과정 중. 관심분야는 VOD, 멀티미디어, Mobile computing

이 철 원

1987년 충남대학교 수학과(이학사). 1989년 중앙대학교 전자계산학과(이학석사). 2001년 아주대학교 컴퓨터공학과 박사과정 수료. 1989년 ~ 1996년 한국전자통신연구원 선임연구원. 1996년 ~ 2000년 한국정보보호센터 선임연구원/과제책임자. 2000년 ~ 현재 ETRI부설 국가보안기술연구소 팀장. 관심분야는 컴퓨터 및 네트워크 보안, 정보통신기반보호, 정보보호시스템 평가기준

최 경 희

1976년 서울대학교 수학교육과 졸업(학사). 1979년 프랑스 그랑데꼴 Enseeiht대학 졸업(석사). 1982년 프랑스 Paul Sabatier대학 정보공학부 졸업(박사). 1982년 ~ 현재 아주대학교 정보통신전문대학원 교수. 관심분야 운영 체제, 분산시스템, 실시간 및 멀티미디어시스템 등

박 승 규

1974년 서울대학교 응용수학과 졸업(학사). 1976년 한국과학기술원 전산학과 졸업(석사). 1982년 프랑스 Institute National Polytechnique de Grenoble 전산학과 졸업(박사). 1976년 ~ 1992년 KIST, KIET, IBM 왓슨 연구소, ETRI 연구위원. 1992년 ~ 현재 아주대학교 정보통신전문대학원 교수. 관심분야는 컴퓨터 구조, 멀티미디어, 실시간시스템, 이동컴퓨터 등

정 기 현

1984년 서강대학교 전자공학과 졸업(학사). 1988년 미국 Ilinois주립대 EECS 졸업(석사). 1990년 미국 Purdue대학 전기전자공학부 졸업(박사). 1991년 ~ 1992년 현대반도체 연구소. 1993년 ~ 현재 아주대학교 전자공학부 교수. 관심분야는 컴퓨터구조, VLSI 설계, 멀티미디어 및 실시간 시스템 등