NS를 이용한 시나리오기반 공격 시뮬레이터 설계 및 구현☆

Design and Implementation of Scenario-based Attack Simulator using NS

최 향 창* 노 봉 남** 이 형 효***
Hyang-Chang Choi Bong-Nam Noh Lee, Hyung Hyo

요 약

일반적으로 네트워크 공격은 단위 공격이 혼합된 시나리오 형태이다. 시나리오 공격은 광범위한 네트워크 환경에서 이루어지기 때문에 공격 범위가 분명하지 않아 공격과 관련 없어 보이는 불분명한 패킷들까지 분석이 요구된다. 이는 공격에 무관한 패킷들까지 분석에 가담시켜 공격 패턴 탐지를 보다 어렵게 하는 요인이다.

본 논문은 시나리오를 갖는 공격에서 공격에 관련된 패킷 분류를 돕는 공격 시나리오 시뮬레이션 시스템을 설계하고 구현한다. 제안된 시스템은 분석대상 네트워크를 시뮬레이터의 가상환경으로 복제하고, 시나리오에 기반을 둔 공격 행위가 포함된 TCPDUMP 패킷을 복제된 가상환경에서 시뮬레이션 할 수 있다. 이 시스템은 보안 관리자들이 공격 시나리오 패턴분석에 유용하게 활용할 수 있을 것이다.

Abstract

Generally, network attacks are based on a scenario composed of a series of single-attacks. As scenario attacks are launched over a wide network environment and their targets are not apparent, it is required to analyze entire packets captured on the network. This method makes it difficult to detect accurate patterns of attacks because it unnecessarily analyzes even packets unrelated to attacks.

In this paper, we design and implement a simulation system for attacks scenario, which helps packet classification connected with attacks. The proposed system constitutes a target network for analysis in a virtual simulation environment, and it simulates dumping TCPDUMP packets including scenario attacks under the constructed virtual environment. We believe that our proposed simulation system will be a useful tool when security administrators perform the analysis of patterns of attack scenarios.

☞ Keyword : Scenario Attack, Network Simulator, IDS, DARPA

1. 서 론

시나리오 공격은 DDoS(Distributed Denial of Service), DRDoS(Distributed Reflection Denial of Service), HttpTunnel, MultiHop 등 다양하다[1, 2]. 이러한 공격은 광범위한 네트워크 환경과 다양한 시스템들을 공격에 이용하기 때문에 단위 IDS(Intrusion Detection System)로는 공격이 시작된 근원지를 찾기 힘들어 공격에 대한 원천 대응이 어렵다[3]. 하지만 시나리오기반 침입탐지 시스템은 여러 단위 침입탐지시스템, 라우터 등 다양한 네트워크 자원들을 서로 통합(federation)하여 시나리오 공격패턴을 실시간으로 공유해 시나리오 공격을 탐지한다. 하지만 시나리오 공격은 광범위한 네트워크 환경에서 이루어지기 때문에 공격범위

* 정 회 원 : 전남대학교 시스템보안연구센터 선임연구원
hcchoi@lsrc.jnu..ac.kr(제1저자)
** 종신회원 : 전남대학교 컴퓨터정보학부 교수
bongnam@chonnam.ac.kr
*** 정 회 원 : 원광대학교 정보・전자상거래학부 조교수
hlee@wonkwang.ac.kr(교신저자)
[2005/11/03 투고 - 2005/11/18 심사 - 2006/06/05 심사완료]
☆ 이 논문은 2005년도 원광대학교의 교비지원에 의해서 수행됨

가 분명하지 않아 공격 분석을 위해서는 공격과 관련되어 보이는 불분명한 패킷들까지 분석이 요구된다[8]. 이것은 시나리오 공격에 무관한 패킷들까지 공격에 가담시켜 시나리오 공격 분석을 어렵게 한다[4, 5].

본 논문은 시나리오 공격 분석의 패턴 추출을 손쉽게 하기위해 공격관련 패킷 분류를 돕는 공격 시나리오 시뮬레이션 시스템을 설계하고 구현한다. 이 시스템은 분석대상 네트워크를 가상환경으로 복제하고, 복제된 환경에서 시나리오에 기반을 둔 공격 행위를 포함하는 TCPDUMP[7] 패킷을 처리하여 패킷 흐름을 시각화 할 수 있다. 따라서 분석자는 광범위한 분석 자료를 제안된 시스템으로부터 시뮬레이션 하여 공격 시나리오 과정을 시각적으로 분석함으로써 공격의 패턴 추출에 유용하게 활용 할 수 있다.

본 논문의 2장은 네트워크 시뮬레이터[6]에서 TCPDUMP 패킷을 시뮬레이션 할 수 있도록 확장하기 위해 NS-2[6] 시뮬레이터를 분석하고, DARPA [1, 2]에서 제공하는 다양한 공격 시나리오를 분석한다. 3장은 제안하는 공격 시나리오 시뮬레이션 시스템을 설계하고, 4장은 설계한 모듈을 구현한다. 또한 DARPA의 DDoS 데이터 셋[2]을 제안한 시스템에서 시뮬레이션 한 결과를 보인다.

2. 관련 연구

2.1 네트워크 시뮬레이터

NS-2는 UCB(University of California, Berkeley)의 LBNL(Lawrence Berkeley National Laboratory)에서 개발 되었다[6]. NS-2는 C++과 OTcl을 기반으로 개발된 객체지향 이산사건 구동 시뮬레이터로 LAN과 WAN, 그리고 TCP/IP를 기반으로 하는 다양한 네트워크를 시뮬레이션 하는데 유용하게 사용된다. NS-2의 동작방법은 NS-2에서 제공하는 문법에 Tcl/Tk를 기반으로 가상 네트워크 환경과 네트워크 자원에 대한 사용 흐름을 프로그램하면 이는 네트워크애니메이터(NAM : Network Animator)에서 시각적으로 시뮬레이션 된다[9]. 이 네트워크애니메이터는 토폴로지 레이아웃과, NS-2에서 프로그래밍 된 결과물을 해석하여 패킷 레벨의 애니메이션을 지원하고, 또한 Xgraph와 같은 추이적인 분석도 제공한다.

NS-2는 네트워크 환경을 가상으로 설정하고 테스트하는 기능은 제공하지만 실제 네트워크(real network)에서 수집된 네트워크 패킷인 TCPDUMP를 프로그램 수준에서 해석하여 시뮬레이션 하지 못한다[6, 9]. 우리는 이러한 문제를 개선하여 공격 패킷이 포함된 TCPDUMP를 NS-2에서 시뮬레이션 할 수 있도록 지원하는 시스템을 개발한다.

2.2 침입시나리오

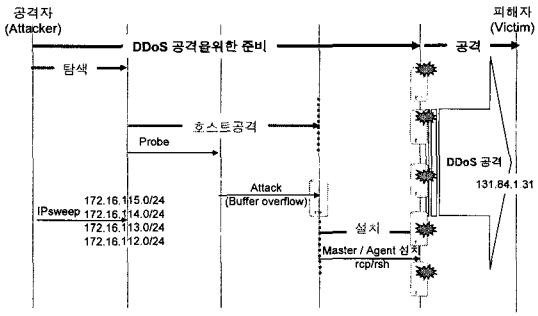
침입탐지 시나리오 중 LINCON LAB의 DARPA 데이터 셋은 광범위하고 다양한 공격의 패킷 데이터 셋을 제공한다[1, 2, 10]. 우리는 DARPA의 데이터 셋을 분석해 몇몇 시나리오 공격을 추출하였다. 이중 분석된 시나리오는 HttpTunnel, MutiHop, SnmpGet, RootKit, DDoS 이다.

HttpTunnel 시나리오는 방화벽이 설치되어있는 내부네트워크에서 외부의 공격자가 중요한 정보를 빼내기 위해 사용하는 시나리오 형태의 공격기법이다. 이 시나리오는 웹 서비스 취약점을 이용해 내부 정보를 도청(sniffing)[11]하는 방법을 제공한다. MultiHop 시나리오는 해커가 공격을 감행할 때 자신의 위치를 노출시키지 않기 위해 여러 컴퓨터들을 경유하는 공격기법이다. 이 공격 시나리오는 인터넷상에 연결된 매우 취약한 컴퓨터들을 공격하여 확보하고 공격의 타겟이 되는 시스템의 취약점을 찾기 위해 확보된 컴퓨터들을 경유하여 조사(Probe)[11]를 감행하고 취약점을 확보한다. 이후 확보한 컴퓨터를 이용하여 다중의 세션을 맺고 공격의 최종목표가 되는 시스템의 중간도구로서 다양하게 활용하는 공격시나리오다.

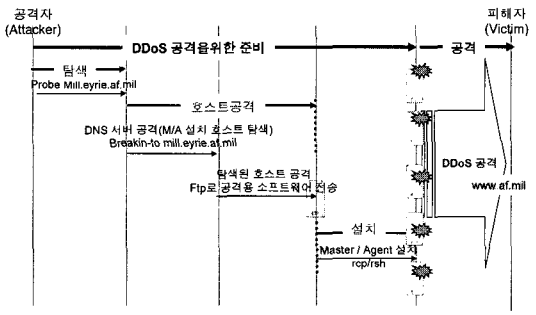
SnmpGet 시나리오는 보안으로 취약한 라우터를 검색하여 이 라우터를 제어하는데 사용되는 비밀번호를 해킹한 후에 해당 라우터를 경유하는 패킷들을 수집하여 분석한다. 이때 수집된 정보는 사용자가 웹사이트나 FTP, Telnet, E-Mail 서비스에 접속하기 위해 필요한 ID와 비밀번호가 있을 수 있다. 이 정보는 제2의 공격으로 이용될 수 있다. RootKit 시나리오는 공격에 관련된 호스트를 침투하여 RootKits인 BackDoor[11]를 설치한 후 이를 이용해 세션을 성립하여 제2의 공격을 감행한다. 이러한 RootKit 공격은 RootKit의 종류에 따라 여러 유형의 공격으로 파생될 수 있다.

분산서비스거부(DDoS : Distributed Denial of Service) 공격 시나리오는 합법적인 사용자 서비스에 대한 정당한 요구를 거부 한다[12, 15]. 공격자(Attacker)가 여러 개의 마스터(Master)를 제어하고 하나의 마스터는 여러 개의 에이전트(Agent)를 제어한다. 매우 많은 수의 에이전트 시스템들이 하나의 공격목표 호스트(Victim)의 서비스 불능을 목표로 수많은 패킷을 전송함으로써 공격을 수행한다[12]. 이러한 DDoS 공격은 5 단계로 구성된다. 1 단계는 IP Sweep[11], 2 단계 Probe, 3 단계는 2 단계에서 Probe한 사이트에 대한 취약점을 공격하여 루트권한을 획득한다. 4 단계는 설치단계로서 루트권한을 획득한 시스템에 마스터나 에이전트를 설치한다. 1~4 단계에 의해 DDoS공격을 위한 준비를 마치고 5단계에서 공격 목표 서비스나 장치를 대상으로 공격자가 마스터에게 명령을 개시한다. 이렇게 되면 마스터에 연결된 수많은 호스트들을 통해 DDoS공격이 이루어진다. DARPA의 DDoS공격 데이터 셋은 두 유형으로 나뉜다[2]. 이중 일반적인 공격 시나리오는 (그림 1)과 같다.

(그림 2)는 스텔스(stealth)한 DDoS공격이 분석된 결과를 보인다.



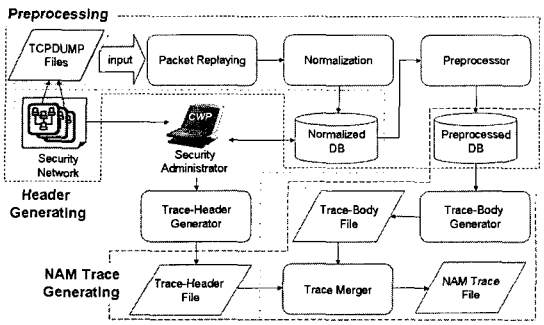
(그림 1) DARPA의 일반 DDoS 공격 셋



(그림 2) DARPA의 스텔스한 DDoS 공격 셋

3. 공격 시나리오 시뮬레이션 시스템 설계

NAM은 이벤트를 기반으로 각각의 트레이스를 패킷의 흐름에 따라 순차적으로 기록한다. 각 이벤트는 속성을 가지고 있으며 이 속성에 기반을 두어 네트워크 토폴로지(topology)와 각 토폴로지에 대한 패킷의 흐름을 시뮬레이션 한다[6]. NAM 애니메이터에서 시뮬레이션 가능한 NAM 트레이스 파일은 네트워크 토폴로지에서 필요한 속성을 정의하는 헤더(header)부분과 패킷의 흐름을 묘사하는 바디(body)부분으로 구성된다[9]. 우리는 NS-2 시뮬레이터를 활용하여 공격행위를 포함하는 TCPDUMP를 NAM에서 시뮬레이션 할 수 있는 시스템을 설계하고 구현한다.



(그림 3) 공격 시나리오 시뮬레이션 시스템 구조

우리가 제안한 공격 시나리오 시뮬레이션 시스템은 (그림 3)과 같이 크게 3단계로 구성된다.

*이중 전처리(Preprocessing) 단계*는 패킷의 헤더 정보에 기반을 두어 수집한 네트워크 패킷덤프를 준비하고 그 패킷덤프에 따른 네트워크 상황에 대한 기본정보를 추출하는 단계이다.

*헤더 생성(Header Generating) 단계*는 *보안 관리자(Security Administrator)*가 패킷을 수집한 네트워크 상황 정보를 받아 네트워크 환경을 분석하고 이 정보를 입력하여 *트레이스헤더 파일(Trace-Header File)*을 생성한다.

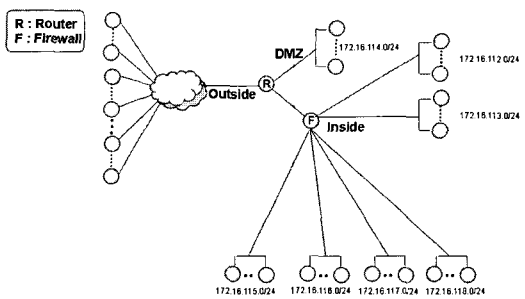
*NAM 트레이스 생성(NAM Trace Generating) 단계*는 *헤더 생성* 단계를 통해 획득한 *트레이스헤더 파일*과 전처리단계를 거쳐서 생성된 *전처리된 DB*를 *트레이스바디 생성기(Trace-Body Generator)*를 통해 *트레이스바디 파일(Trace-Body File)*을 생성하고, 이 둘을 *트레이스 통합 (Trace Merger)* 모듈이 입력 값으로 받아 *NAM 트레이스 파일(NAM Trace File)*을 생성한다.

3.1 단계 1: 전처리(Preprocessing)

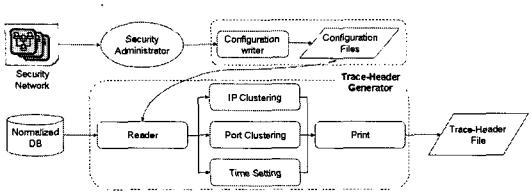
전처리 단계는 *Packet Replaying, Normalization, Preprocessor* 모듈들로 구성된다. 이 단계는 다양한 *보호 네트워크(Security Network)*로부터 수집된 *TCPDUMP[7] 파일(TCPDUMP Files)*들을 입력으로 받아 LibPcap[13]을 사용하여 수집된 패킷덤프들을 서로 다른 네트워크 환경에 시간 값을 동기화 시켜 재생한다. 재생되는 패킷들은 정규화(Normalization)모듈을 통해 IPv4헤더정보에 데이터 사이즈의 크기만을 결합하여 축약된 *정규화된 DB(Normalized DB)*를 생성한다. 표준화된 DB는 *전처리기(Preprocessor)*를 통해 시간, 출발지 및 목적지 IP, 식별자(identification), 데이터 크기의 정보 구조를 갖는 *전처리된 DB(Preprocessed DB)*로 변환된다. TCPDUMP에서 추출되는 데이터 필드는 패킷의 유형, 날짜/시간, 패킷에 포함된 데이터의 크기, 출발지 주소, 목적지 주소로 구성된다. 패킷 하나의 구분은 줄 단위로 구분되며 각 정보들은 ','로 구분되어 *Preprocessed DB*에 저장 된다.

3.2 단계 2: 헤더 생성(Header Generating)

(그림 3)의 *Security Administrator*는 *Trace-Header File* 생성을 위해서 보호대상 네트워크 환경에 대한 분석이 요구된다. 예를 들어 DARPA의 DDoS 데이터 셋[7]은 크게 DMZ 영역과 내부(inside) 영역으로 이루어져 있다. 이 각각의 DDoS 덤프(dump)파일의 내용에서 IP를 추출하고 그에 따라 클래스 형태로의 분할을 수행했다. C 클래스 형태로 그룹화 하면 크게 7개의 서브 클래스를 얻을 수 있다. (그림 4)는 이와 같은 과정으로 DARPA 데이터 셋에서 DDoS용 TCPDUMP를 분석하여 그려낸 네트워크 토폴로지(network



(그림 4) DARPA의 일반 DDoS 공격의 네트워크 토폴로지



(그림 5) 헤더 생성 세부모듈의 동작순서

topology)이다. (그림 4)의 네트워크 토폴로지는 DARPA데이터 셋으로부터 추출된 것이지만 실제의 환경에서는 네트워크 토폴로지를 보안 관리자가 보호대상 네트워크의 네트워크 관리자(network administrator)로부터 획득할 수 있다.

(그림 5)는 *헤더 생성*의 세부모듈의 동작순서를 보인다. 이중 *CWP(Configuration Writing Procedure)*는 (그림 8)의 헤더생성 단계의 *Security Administrator*가 동작시키는 응용이다. *Trace-Header Generator*의 세부모듈들을 수행시키기 이전에는 보안 관리자가 *Configuration writer* 모듈을 이용하여 *Configuration Files*을 생성한다. 이 *Configuration Files*은 보안 관리자에 의해 분석된 토폴로지인 (그림 4)를 *Configuration writer*모듈의 인터페이스를 이용해 숫자와 기호의 조합으로 재표현한 것이다. 이는 다음 장에서 보이는 NAM 트레이스 생성기에 입력된 각각의 데이터들로 표현된다. *Trace-Header Generator*의 세부 모듈 중 *Reader* 모듈은 *Normalized DB*와 *Configuration Files*을 입력 받아서 각 정보를 XACML의 SAX파서를 이용하여 파싱하고 *IP Clustering, Port Clustering, Time Setting* 세부모듈을 통해 (그림 6)과 같은 헤더 데이터 구조형태로 데이터를 유지한다. (그림 6)의 헤더 생성 데이터 구조에서 lan [] 은 랜(lan)의 ID로 이루어진 배열이며 gate [] 는 게이트웨이(gateway)의 역할을 하는 노드(node)의 배열이다. 또한 sub_mask [] [18]은 lan [] 과 동기화하여 해당 랜의 서브넷마스크(subnet mask)를 저장한다. 그리고 max_node는 현 토폴로지(topology)의 총 노드 수를 의미한다. route_table [] [] 은 노드 ID 값으로 연결된 라우팅(routing) 정보, addr_list [] [15]는 '.'으

```
struct topology {
        int lan[];
        int gate[];
        char sub_mask[][18];
        int max_node;
};

int route_table[][];
char addr_list[][15];
int addr_map[][2];
```

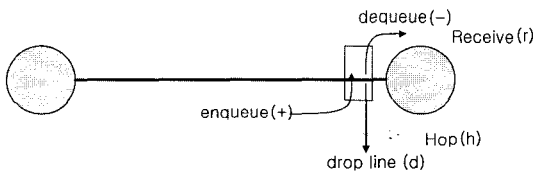
(그림 6) 헤더 생성 데이터 구조

로 구분된 십진 IP 주소(decimal IP address), addr_map [] [2]은 addr_list [] [15]와 동기화하여 IP 주소(IP address)에 해당하는 노드 ID(node ID)와 서브넷 ID(subnet ID)를 저장한다[14].

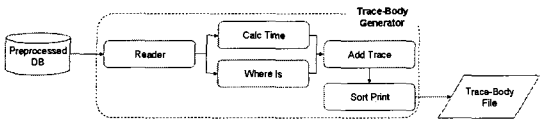
3.3 단계 3: NAM 트레이스 파일 생성(NAM Trace Generating)

*NAM 트레이스 파일 생성 단계*는 *Trace-Body Generator*와 *Trace Merger* 모듈로 구성된다. *NAM 트레이스 파일 생성 단계*는 *Preprocessed DB*와 *Trace-Header File*을 이용해서 *NAM Trace File*을 생성한다. 하나의 TCP/IP 패킷은 *NAM Trace File*에서 (그림 7)의 각 상황을 고려하여 라우팅 홉마다 여러 개의 이벤트로 나누어서 표현 된다.

(그림 7)과 같이 하나의 패킷은 *NAM Trace File*에서 패킷이 생성되어서 소멸될 때까지의 과정 모두를 묘사할 수 있도록 각각의 상황에 대해 표현해야 한다. *NAM Trace File*은 패킷의 라이프사이클 동안 그 패킷에게 주어졌던 각각의 상황을 이벤트화 하고 그것에 대해 표현이 가능하도록 h, +, -, r, d 플래그(flag)를 제공한다[6].



(그림 7) NS-2의 이벤트 상황



(그림 8) 트레이스-바디 생성기 세부 모듈

*트레이스-바디 생성기*는 *Preprocessed DB*를 패킷 단위로 읽고 *NAM Trace File* 형태로 변환하여 기록한다. *NAM Trace File*의 생성에 필요한 세부 모듈은 (그림 8)과 같다.

Reader 모듈은 *Preprocessed DB*의 한 라인을 읽어 들여 목적지 IP, 출발지 IP, 프로토콜(protocol), 시간(time), 패킷 ID, 패킷 크기를 읽어낸다.

*Calc Time*모듈은 순차적으로 *NAM Trace File*을 유지할 수 있도록 시간 값을 계산하여 처리하는 모듈이다. 이 모듈은 TCPDUMP의 시간 값을 기준으로해서 시간 값을 계산하여 TCPDUMP에서 획득한 패킷덤프의 시간 동기화를 수행한다. 이것은 시뮬레이션 할 TCPDUMP 각각의 시간을 행 단위로 읽어 가장 빠른 시간을 시작시간(0: 나노초 단위)으로 잡고, 이 패킷을 기준으로 상대적인 시간 값을 산정하여 처리한다. 즉, 상대적으로 증가시켜 부여되는 값은 0 나노초부터의 순차적인 *NAM Trace File*이 생성된다. 또한 이렇게 생성된 *NAM Trace File*을 NAM 애니메이터에서 시뮬레이션하기위해 충분한 시간간격이 될 수 있도록 보안 관리자가 설정할 수 있는 임의의 상수(k) 값을 두어 조절한다. *Calc Time*모듈의 동작을 예로 들면 (그림 9)와 같은 패킷이 있을 경우 (그림 10)과 같이 첫 시간인 952438896.476837은 0.0으로 세팅하고 다음 패킷의 시작시간인 952438896.477263은 첫 시간과의 차이인 0.000426이 NAM에서의 시간 값이 된다. 하지만 NAM 애니메이터는 기본적으로 나노초 단위에서 트래픽(traffic)표현이 가장 이상적이므로 시간 값은 0.000043으로 자동 계산한다[6].

time	size	id	src_addr	dst_addr
952438896.476837	41	41213	172.16.113.168	172.16.112.50
952438896.477263	41	37056	172.16.112.50	172.16.113.168
		... ...		
		... ...		

(그림 9) Tcpdump 패킷 표현방식

```
h -t 0.000000 -s 16 -d 47 -p tcp -e 41 -c 1 -i 41213 -a 1
h -t 0.000043 -s 7  -d 46 -p tcp -e 41 -c 1 -i 37056 -a 1
                    ……
                    ……
```

(그림 10) NAM 트레이스파일 표현 방식

Where Is 모듈은 *Reader* 모듈에서 읽어 들인 IP 주소를 네트워크 토폴로지 정보와 비교하여 읽은 IP에 해당되는 노드(node id), 서브넷(subnet) 등을 찾아낸다.

Add Trace 모듈은 네트워크 토폴로지 정보의 라우팅 테이블과 정해진 알고리즘에 따라 *NAM Trace File*을 어떻게 생성 할지를 결정한다[14]. 라우팅 테이블은 출발지에서 목적지로 가는 경로상의 모든 노드나 lan을 포함하고 있고 이를 바탕으로 node_routing, lan_routing 유무를 결정하고 각각의 알고리즘을 적용하여 트레이스 파일을 생성하고 홉(hop)을 지날 때마다 Calc Time 모듈에서 계산된 값에 따라 시간 값을 증가 시킨다. *NAM Trace File*에서 패킷 흐름의 표현은 노드 사이의 링크를 지날 때와 랜(lan)사이에서의 트래픽(traffic)을 표현하는 두 방식이 있다. node_routing은 (그림 11)과 같이 노드와 노드사이의 패킷(packet)흐름을 기반으로 홉(hop) 경로와 들어오는 패킷(enqueue), 나가는 패킷(dequeue)을 수행하는 과정을 생성한다. lan_routing은 (그림 12)와 같이 랜의 한 호스트에서 다른 호스트로 갈 때 일반적으로 방송되어 나가고 목적지 외의 다른 호스트에서는 자신이 받은 패킷을 버리는 과정이 추가적으로 필요하다[6].

lan_routing의 다양한 토폴로지(topology)를 가정하여 상황에 따라 트레이스(trace)를 생성한다. 예를 들어 (그림 13)에서 node 1에서 node 2로의

```
h -t 시작시간 -s 출발지 -d 도착지 -p 프로토콜 -e 크기 -c 1 -i 패킷아이디 -a 1 -x
(시작시간+0.001)
+ -t 시간 -s 출발지 -d 도착지 -p 프로토콜 -e 크기 -c 1 -i 패킷아이디 -a 1 -x
- -t 시간 -s 출발지 -d 도착지 -p 프로토콜 -e 크기 -c 1 -i 패킷아이디 -a 1 -x
h -t 시간 -s 출발지 -d 도착지 -p 프로토콜 -e 크기 -c 1 -i 패킷아이디 -a 1 -x
```

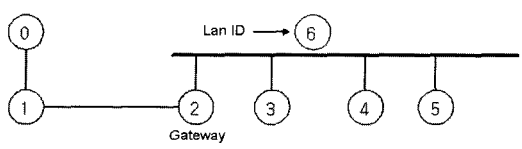
(그림 11) NAM 트레이스파일에서 일반링크의 패킷 흐름 표현

```
h -t 시작시간 -s 출발지 -d 랜번호 -p 프로토콜 -e 크기 -c 1 -i 패킷아이디 -a 1 -x
(시작시간+0.001)
+ -t 시간 -s 출발지 -d 랜번호 -p 프로토콜 -e 크기 -c 1 -i 패킷아이디 -a 1 -x
- -t 시간 -s 출발지 -d 랜번호 -p 프로토콜 -e 크기 -c 1 -i 패킷아이디 -a 1 -x
h -t 시간 -s 출발지 -d 랜번호 -p 프로토콜 -e 크기 -c 1 -i 패킷아이디 -a 1 -x
(시간+0.001)
for ( i = 0 ; i < number of lan's node ; i++ ) {
  d -t 시간 -s 도착지 외의 노드 번호 -d 랜번호 -p 프로토콜
     -e 크기 -c 1 -i 패킷아이디 -a 1 -x
       }
(시간+0.000004)
r -t 시간 -s 도착지 번호 -d 랜번호 -p 프로토콜 -e 크기 -c 1 -i 패킷아이디 -a 1 -x
(시간+0.000004)
```

(그림 12) NAM 트레이스파일에서 랜의 패킷 흐름 표현

packet흐름을 node_routing으로 표현할 수 있지만 node 1에서 랜(lan)의 호스트인 node 3으로의 흐름은 node_routing으로 게이트웨이(gateway)인 node 2까지 간 후 node 2에서 node 3으로 lan_routing이 이루어지게 된다. 즉 *NAM Trace File*은 (그림 14)의 유형에 의해 생성된다.

*NAM Trace File*은 시간 값에 따른 순차적인 파일구조를 가져야 한다. 하지만 한 패킷을 읽고 그에 해당하는 라우팅 경로에 따라 트레이스 파일을 기록하게 되면, 패킷의 마지막 이벤트에서의 시간 값은 다음 패킷의 처음 이벤트의 시간 값 보다 늦는 경우가 생기게 된다. *Sort Print*모듈은 생성된 트레이스 각각을 순차적인 시간 값을 기준으로 정렬을 수행한다.



(그림 13) 하나의 랜에 연결된 노드 토폴로지

```
case1 : node → node
  node_routing(node, another node)

case2 : node → lan's host
  node_routing(node, lan's gateway)
  lan_routing(lan's gateway, lan's host)

case3 : node → lan's gateway
  node_routing(node, lan's gateway)

case4 : lan's host → node
  lan_routing(lan's host, lan's gateway)
  node_routing(lan's gateway, node)

case5 : lan's host → another lan's host
  lan_routing(lan's host, lan's gateway)

case6 : lan's host → lan's gateway
  lan_routing(lan's host, lan's gateway)

case7 : lan's gateway → lan's host
  lan_routing(lan's gateway, lan's host)

case8 : lan's gateway → node
  node_routing(lan's gateway, node)
```

(그림 14) NAM 트레이스 파일의 8가지 유형

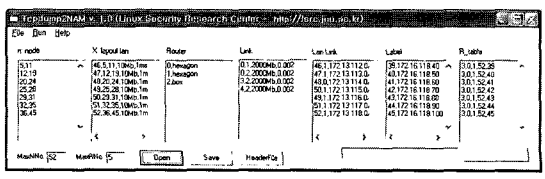
4. 시스템 구현 및 DDoS 시나리오 시뮬레이션

시스템은 3장의 설계에 기반을 두어 c, pascal 언어로 구현하였다. 구현된 결과를 테스트 하기위해 2장에서 분석된 DDoS 공격 시나리오를 시뮬레이션 한다.

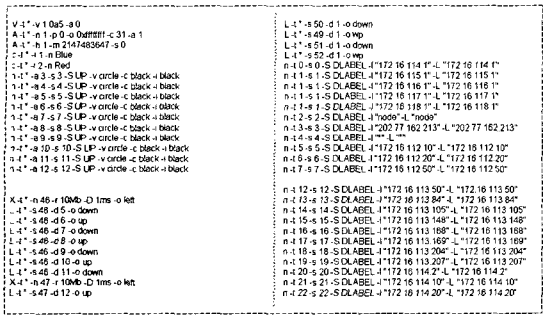
4.1 NAM 트레이스 파일 생성기

(그림 15)는 DDoS 시나리오를 시뮬레이션하기 위해 분석자가 기본정보를 입력한 *NAM 트레이스 파일 생성기(NAM Trace File APP)*를 나타낸다.

NAM 트레이스 파일 생성기에서 'Open 버튼'은 네트워크 관리자의 도움을 받아 노드, 랜, 라우터,



(그림 15) NAM 트레이스 파일 생성기



(그림 16) NAM 트레이스 헤더 파일 구조

링크, 랜 연결설정, 레이블을 설정한 파일을 읽어 들인다. 이렇게 읽어 들인 파일은 'HederFile' 버튼'으로 (그림 3)의 *Trace-Header Generator* 모듈을 통해 NAM 애니메이터에서 해석할 수 있는 (그림 16)과 같은 NAM 트레이스 헤더 파일을 생성한다. 이 트레이스 헤더 파일은 노드들의 연결정보, 노드 색깔이나 모양과 같은 시각정보, 링크의 연결 속도나 큐잉 방법들이 기술된다.

이후 'NamFile버튼'을 클릭하면 분석할 TCPDUMP를 읽어 들여 처리하여 생성하는 (그림 3)의 *NAM Trace Generating* 단계의 *Trace-Body Generator*를 거쳐 (그림 16)과 같은 NAM 트레이스 바디 파일(*Trace-Body File*)을 발생시간에 기준해서 순차적으로 생성한다.

끝으로 (그림 16)과 (그림 17)을 *NAM Trace Generating* 단계의 *Trace Merger*를 거쳐 단일한 NAM 트레이스 파일(*NAM Trace File*)로 통합하고 이것을 NAM 네트워크애니메이터에 넘긴다.

```
h -t 0.000000 -s 16 -d 47 -p tcp -e 41 -c 1 -i 41213 -a 1
+ -t 0.001000 -s 16 -d 47 -p tcp -e 41 -c 1 -i 41213 -a 1
- -t 0.001000 -s 16 -d 47 -p tcp -e 41 -c 1 -i 41213 -a 1
h -t 0.001000 -s 16 -d 47 -p tcp -e 41 -c 1 -i 41213 -a 1
d -t 0.001004 -s 12 -d 47 -p tcp -e 41 -c 1 -i 41213 -a 1
d -t 0.001004 -s 13 -d 47 -p tcp -e 41 -c 1 -i 41213 -a 1
d -t 0.001004 -s 14 -d 47 -p tcp -e 41 -c 1 -i 41213 -a 1
d -t 0.001004 -s 15 -d 47 -p tcp -e 41 -c 1 -i 41213 -a 1
d -t 0.001004 -s 16 -d 47 -p tcp -e 41 -c 1 -i 41213 -a 1
d -t 0.001004 -s 17 -d 47 -p tcp -e 41 -c 1 -i 41213 -a 1
d -t 0.001004 -s 18 -d 47 -p tcp -e 41 -c 1 -i 41213 -a 1
d -t 0.001004 -s 19 -d 47 -p tcp -e 41 -c 1 -i 41213 -a 1
r -t 0.002004 -s 47 -d 1 -p tcp -e 41 -c 1 -i 41213 -a 1
h -t 0.003004 -s 16 -d 47 -p tcp -e 41 -c 1 -i 41213 -a 1
                    ... ...
                    ... ...
```

(그림 17) NAM 트레이스 파일

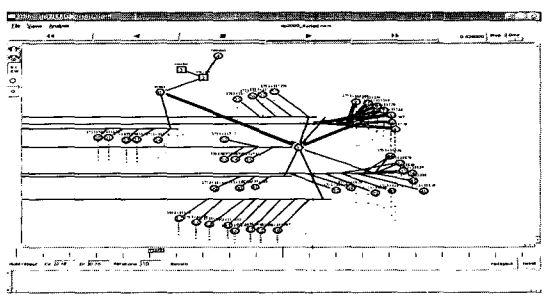
4.2 수행결과

4.1로부터 넘겨받은 파일을 통해 NAM 네트워크 애니메이터가 실행되고 시뮬레이션이 진행된다. (그림 18)은 DARPA 의 DDoS 공격 데이터 셋의 시나리오 중 5단계를 시뮬레이션 한 결과를 보인다.

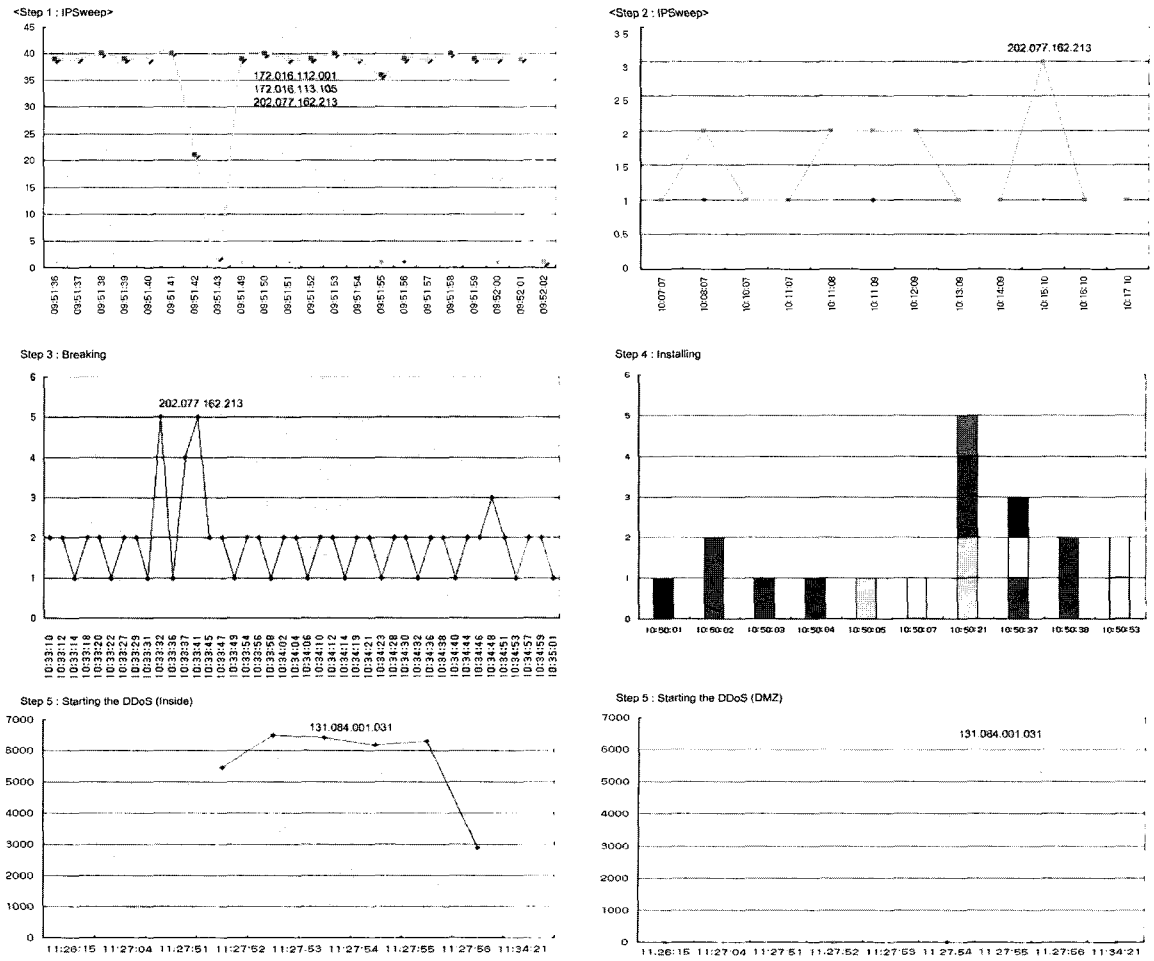
우리가 제안한 도구를 이용해 보안 관리자는 공격이 포함된 TCPDUMP를 분석하여 쉽게 DDoS 공격으로 감지했다. 이후 (그림 1)과 같이 사전 분석된 DDoS 시나리오를 기준으로 각 단계별로 패킷들을 클러스터링(clustering) 했다. (그림 19)는 이후 분석된 DDoS의 특성을 보인다.

(그림 19)에서 단계 1(Step 1)은 IPSweep의 대상 타겟(target) 별로 통계치를 산정하였다.

이는 IP Sweep공격을 받는 대상인 IP들이 초당 40개정도의 패킷을 받아냈음을 보인다. 또한 이것으로 우리는 IP Sweep공격을 받은 호스트들을 예상할 수 있다. 단계 2(Step 2)는 Probe공격을 받은 IP를 발견할 수 있다. 즉 이 호스트가 DDoS의 공격에 가담시킬 응용을 설치할 타겟으로 SunRPC/udp 포트인 111번을 공격받았음을 보인다. 단계 3(Step 3)은 Breaking 공격이며 단계 2와 유사하다. 단계 4(Step 4)는 Installing 공격으로 텔넷(telnet)과 쉘(shell) 포트에 관련하여 (그림 19)와 같은 특징을 보인다. 단계 5(Step 5)는 DDoS공격의 개시로 INSIDE와 DMZ에서 공격대상 호스트에 초당 6000개 이상의 패킷 범람(packet flooding)의 특징을 보인다. 위와 같이 분석된 결과는



(그림 18) DARPA 2000 데이터 셋 시뮬레이션 화면



(그림 19) 분석된 DDoS 공격특징

DDoS 시나리오를 탐지하는데 충분한 특성으로 활용될 수 있을 것이다.

5. 결 론

본 논문에서는 보안 관리자나 네트워크 관리자가 분산 서비스 거부공격과 같은 침입행위를 쉽게 판별할 수 있도록 네트워크 시뮬레이터의 일종인 NS-2의 NAM을 통해 TCPDUMP 네트워크 패킷을 시뮬레이션 하는 시스템을 제안하고 이를 이용해 DARPA의 2000년도 DDoS 데이터 셋[2]을 시뮬레이션 했다.

관리자는 제안된 공격시나리오 시뮬레이션 시스템을 통해 시각화된 실제 네트워크 상황을 볼 수 있고 공격을 판단하는데 많은 도움을 받을 수 있을 것이다. 하지만 제안된 시스템을 위해서는 시뮬레이션 네트워크 위상이나 시스템 입력 포맷, TCPDUMP 등과 같은 많은 사전 지식을 필요로 한다. 다음 연구에서는, 네트워크 위상을 자동으로 설계하여 네트워크 분석자의 참여를 줄여 시스템의 편의성을 높이고, 시나리오기반 침입탐지 시스템과의 연계를 통해 시나리오 공격에 대응할 수 있는 시스템으로 발전시키고자 한다.

참 고 문 헌

[1] MIT Lincoln Lab, "1999 DARPA Intrusion Detection Scenario Specific Datasets," LINCOLN LABORATORY, 1999, Available from http://www.ll.mit.edu/IST/ideval/data/1999/1999_data_index.html

[2] MIT Lincoln Lab, "2000 DARPA inturusion detection scenario specific datasets," LINCOLN LABORATORY, 2000. Available from http://www.ll.mit. edu/IST/ ideval/data/2000/LLS_DDOS_1.0.html

[3] R. Bace, "Intrusion Detection," Macmillan Technolgy Publishing, 2000.

[4] P. Ning, Y. Cui, D. S. Reeves, "Constructing Attack Scenarios through Correlation of Intrusion alerts," 9th ACM conference on computer and communications security, pp. 245-254, 2002.

[5] Steven Noel, Eric Robertson, Sushil Jajodia, "Correlating Intrusion Events and Building Attack Scenarios Through Attack Graph Distances," ACSAC'04, 2004.

[6] Kevin Fall and Kannan Varadhan, "NS notes and documentation," The VINT Project, UC Berkeley, LBL, USC/ISI, and Xerox PARC, November 1997, Available from http://www-mash.cs.berkeley.edu/ns/

[7] V. Jacobson, C. Leres, and S. McCanne, "TCPDUMP," Lawrence Berkeley National Labs Network Research Group, June, 1989, Available from http://www.tcpdump. org/

[8] F. Cuppens, A. Miege, "Alert Correlation in a Cooperative Intrusion Detection Framework," In Proc, of the 2002 IEEE Symposium on Security and Privacy, 2002.

[9] Paul Meeneghan, Declan Delaney, "An Introduction to NS, Nam and OTcl scripting," National University of Ireland, Maynooth NUIM-CS-TR-2004-05, 2004

[10] Joshua Haines, Lee Rossey, Rich Lippmann and Robert Cunnigham, "Extending the 1999 Evaluation", In the Proceedings of DISCEX 2001, June 11-12, 2001

[11] Joel Scambray, Stuart McClure, Georage Kurtz, "HACKING EXPOSED: Second Edition," McGraw-Hill, 2001

[12] Ratul Mahajan, Steven M. Bellovin, Sally Floyd, John Ioannidis, Vern Paxson, and Scott Shenker, "Controlling High Bandwidth Aggregates in the Network," Computer Communications Review, July 2002.

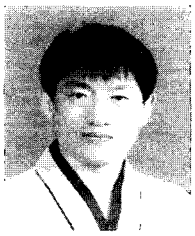
[13] V. Jacobson, C. Leres, and S. McCanne, "libpcap," Lawrence Berkeley National Labs Network Research Group, June, 1989, Available from http://www.tcpdump.org/

[14] George F. Riley, Dheeraj Reddly, "Simulating REalistic Packet Routing Without Routing Protocols," Proceedings of the Workshop on Principles of Advanced and Distributed Simulation, 2005

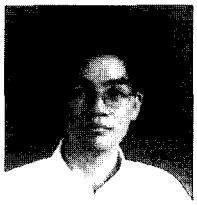
[15] J. Ioannidis and S. Bellovin. Implementing Pushback: Router-Based Defense Against DDoS Attacks. In Proceedings of NDSS '02, Feb. 2002.

◐ 저 자 소 개 ◑

최 향 창
2002년 전남대학교 전산학과 졸업(석사)
2005년 전남대학교 정보보호협동과정 졸업(박사)
2001년~현재 전남대학교 시스템보안연구센터 선임연구원
관심분야 : 침입탐지 시스템, 유비쿼터스 보안, 프라이버시 보호
E-mail : hcchoi@lsrc.jnu..ac.kr



노 봉 남
1978년 2월 전남대학교 수학교육과 졸업(학사)
1982년 2월 KAIST 대학원 전산학과 졸업(석사)
1994년 2월 전북대학교 대학원 전산과 졸업(박사)
1983년~현재 전남대학교 컴퓨터정보학부 교수
2000년~현재 시스템 보안 연구센터 소장
관심분야 : 컴퓨터와 네트워크 보안, 정보보호시스템, 전자상거래 보안, 사이버사회와 윤리
E-mail : bongnam@chonnam.ac.kr



이 형 효
1987년 전남대학교 전산학과 졸업(학사)
1989년 한국과학기술원 전산학과 졸업(석사)
2000년 전남대학교 대학원 전산학과 졸업(박사)
2001~현재 원광대학교 정보·전자상거래학부 조교수
관심분야 : 보안정책 및 보안모델, 유비쿼터스 보안, 프라이버시 보호기술
E-mail : hlee@wonkwang.ac.kr