멀티 쓰레드 기반 N-IDS 모델의 설계 및 구현

주수홍, 엄윤섭, 김상철, 홍승표, 이재호
경성대학교 멀티미디어대학원 정보공학과

A Design and Implementation of N-IDS Model based on Multi-Thread

Jue Sue Hong, Um Youn Sup, Kim Sang Chul, Hong Seung Pyo, Lee Jae Ho
Graduate School of Multimedia, Kyungsung University
Dept. of Information Engineering
E-mail : flstz@orgio.net

요 약

많은 해킹 기법의 발전과 해킹 툴 들이 대중화됨으로 인해 기존의 보안 기술만으로는 발전하는 해킹 기술에 대한 문제를 해결할 수 없게 되었다. 이러한 기존 보안 기술을 대체하기 위한 여러 기술이 등장하였는데 IDS가 그 대표적인 기술 중 하나이다. 네트워크 기반 침입 탐지 시스템인 N-IDS는 패킷에서 침입을 탐지하는 실시간 시스템이다. 따라서 패킷을 캡쳐하고 처리하는 능력이 시스템의 성능을 결정하게 되는데 기존의 N-IDS는 그 구조상 패킷의 캡쳐, 처리 후 다음 패킷 캡쳐까지의 시간 지연이 처리할 패킷의 종류에 따라 불규칙하게 발생한다. 기존의 단일 프로세서 구조를 가진 N-IDS로는 불규칙적인 시간 지연 문제를 해결할 수 없으므로 본 논문에서는 파일 소켓 및 멀티 쓰레드 구조를 이용하여 이러한 문제점을 해결하였다.

ABSTRACT

A network based intrusion detection system(N-IDS), can detect intruders coming in through packets in real time environment. The ability of capture of packet is the most important factor when we evaluate the performance of the system. The time delay between the time handling one packet capture and next one is variant become of packet handling mechanism. So for N-IDS can not settle this problem because most systems use a single processor. In this thesis, we solve the problem of irregular time delay with a file socket and multi-thread processing. We designed and implement, the Crasto system. By an accurate observation, the performance testing shows that the Crasto reduces the capture delay time to 1/5 comparing to the existing single process N-IDS, and maintain delay time regularly.

키워드
IDS, N-IDS, Network Security, Multi Thread

1. 서 론

침입 탐지 시스템은 최근 활발히 연구되고 있는 보안 시스템의 하나로서 시스템 또는 네트워크에서 일어나는 이벤트를 감시하고 침입 여부를 파악하기 위해 이벤트들을 분석하여 실시간으로 침입에 대응하는 보안 시스템이다[2, 4]. 침입 탐지 시스템은 탐지 대상에 따라 호스트 기반 침입 탐지 시스템(Host based IDS: H-IDS)과 네트워크 기반 침입 탐지 시스템(Network based IDS: N-IDS)으로 분류된다[5, 6].

N-IDS는 실시간 시스템으로서 패킷의 처리 능력이 시스템의 성능을 결정하게 된다[4]. N-IDS의 구조는 일반적으로 순차적인 처리 구조를 가진다. 이러한 구조는 한 패킷을 캡쳐하여 처리한 후, 다음 패킷을 처리하는데 일정시간 이상의 시간 지연을 필요로 한다. 그리고 처리하는 패킷의 종류에 따라 지연 시간이 불규칙적으로 발생하게 되며 탐

지성능 향상을 위해 탐지 모듈과 signature가 추가되면 그에 비례하여 지연 시간은 계속해서 증가하게 된다.

이러한 구조적 문제점을 해결하기 위해 본 논문은 기존의 N-IDS 모델을 재 설계하여 멀티 쓰레드와 파일 소켓 구조를 지니는 멀티 쓰레드 기반 N-IDS인 Crasto를 설계 및 구현하였다.

2. 관련 연구

2.1 N-IDS의 기본 계층 구조

그림 1은 일반적인 N-IDS의 기본 계층 구조를 보여준다.

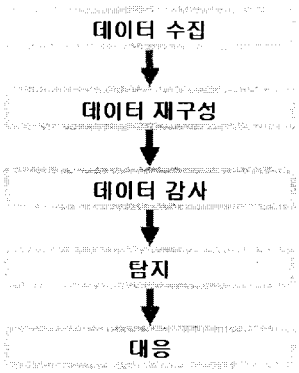

그림 4. N-IDS의 기본 계층 구조

각 계층을 살펴보면 다음과 같다.

- **데이터 수집**

NIC로부터 데이터를 수집하는 부분으로, 시스템의 성능에 크게 영향을 미치는 부분이다[8]. 많은 패킷이 들어올 경우 놓치지 않고 잘 수집할 수 있는 방법이 필요한데, 일반적으로 Snort[9]의 경우 Pcap 라이브러리(Packet Capture Library)[10]를 사용한다.

- **데이터 재구성**

패킷을 원래의 상태로 복원하는 과정으로 정확한 탐지를 위해 반드시 필요한 부분이다.

- **데이터 감사**

재 조합된 패킷이 정상적인 패킷인지 판별한다. 일반적으로 패턴에 의한 공격이 아니고 스캔이나 DoS(Denial of Service) 공격의 경우 여기에서 탐지가 이루어진다. IP 주소나 포트 번호의 이상 유무, 각 프로토콜의 옵션이나 플래그 설정 값의 이상 유무를 검사한다.

- **탐지**

주어진 규칙에 대해서 위반 사항의 유무 등을 확인하는 부분이다. signature 기반 시스템인 Snort의 경우 패킷의 데이터 정보가 동일한 패턴을 가지고 있는지 검사한다. 학습에 의한 탐지도 여기서 수행된다.

- **대응**

침입이 탐지되었을 경우 대응하는 방법이 설정되어 있는 부분이다. 위험도에 따른 로그, 역추적, 관리자의 호출, 경보 등의 기능이 이곳에서 수행된다.

2.2 N-IDS 구성 요소

IDS에서 침입을 탐지하기 위한 대상을 원시 자료(Data Source)[1]라 부르는 데, N-IDS에서는 네트워크 트래픽(패킷)이 원시 자료이다. 이러한 자료를 처리하는 구성 요소는 다음과 같다[1, 3, 4].

- 전처리기(Preprocessor)
- 지식 기반(Knowledge Base)
- 탐지 엔진(Detection Engine)
- 판단 엔진(Decision Engine)

IDS는 지식 기반을 사용하는 시스템이다. 오용 탐지를 위한 N-IDS의 구성 요소는 그림 2와 같다. N-IDS는 구현 시 시스템 독립적인 언어로 침입 유형을 표현 가능하다는 이점 때문에 보통 signature를 기반으로 많이 구현한다[4, 7].

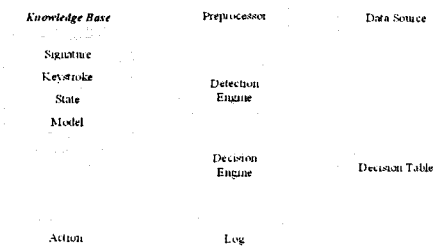

그림 5. 오용 탐지를 위한 N-IDS의 구조

2.3 구조 연구

이 절에서는 N-IDS의 구성 요소들 사이의 연계된 구조에 대해서 알아보고, 현재의 N-IDS가 가지는 구조적 문제점을 알아본다.

2.3.1 linked list structure

N-IDS는 function linked list와 signature linked list 구조를 가진다.

- **Function Linked List(FLL)**

FLL은 function의 관리를 위해 목적이나 기능에 따라 분류하여 생성해 놓은 자료 구조이다. FLL에는 패킷 전처리를 하기 위한 함수 그룹, 침입을 탐지하기 위한 함수 그룹, 침입에 대응하기 위한 함수 그룹이 있다. FLL를 사용하면 각 함수의 추가나 제거가 시스템 작동 중에 가능해지는 이점이 있다.

• **Signature Linked List(SLL)**

SLL은 각 탐지 엔진에서 사용되는 형태에 맞게 분류하여 생성해 놓은 자료 구조이다. SLL의 구성은 일반적으로 signature를 사용하는 많은 침입 탐지 시스템들이 채택하는 구조이다. SLL을 사용함으로 해서 시스템 운영 중에 signature의 관리가 가능해 진다.

2.3.2 loop structure

일반적인 N-IDS 모델은 그림 3과 같은 루프 구조를 가진다. 그림 3에서 한번의 패킷 캡쳐 모듈이 수행되고 나면, 다음 Packet Capture Module (이하 PCM) 수행까지는 다른 모듈의 수행으로 인해 약간의 지연 시간을 가지게 된다. 이러한 지연 시간은 패킷의 종류에 따라 수행되는 모듈의 수와 각 모듈의 수행 시간에 따라 증감하는 특징을 지닌다.

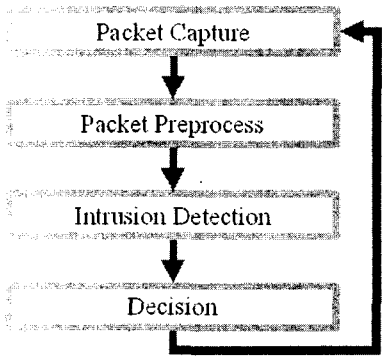

그림 6. 일반적인 N-IDS 모델의 구조

2.3.3 구조적 문제점

FLL, SLL과 loop structure의 구조는 시스템을 운영할 때 유연성과 어느 정도의 속도를 보장해 주는 장점을 지니고 있다. 그러나 loop structure는 구조적 특성으로 캡쳐 지연 시간이 발생시키고, FLL과 SLL은 원소 개수가 증가하면 전체적으로 속도가 떨어지게 된다.

위의 상황을 종합해 볼 때 FLL과 SLL의 원소 증가는 패킷 캡쳐 모듈을 제외한 모든 모듈의 수행 시간을 증가시키게 되며, 수행 시간의 증가는 loop structure에 의해 캡쳐 지연 시간의 증가로 이어지게 된다. 따라서 탐지 함수, 대응 함수, signature의 증가는 패킷 캡쳐 지연 시간을 증가시키며 결과적으로 실시간 시스템인 N-IDS의 성능을 떨어뜨린다. 본 논문은 멀티 쓰레드를 이용하여 패킷 지연 시간 문제를 해결한 멀티 쓰레드 기반 N-IDS인 Crasto를 설계 및 구현한다.

3. 멀티 쓰레드 기반 N-IDS 모델의 설계

3.1 멀티 쓰레드 기반 N-IDS 모델

공개된 N-IDS인 Snort의 구조는 그림 4와 같이 Packet Capture Module(이하 PCM)에서 시작해서 Packet Preprocessor Module(이하 PPM), Intrusion Detection Module(이하 IDM), Decision Module(이하 DM)이 끝난 후 다시 PCM 단계로 이어진다. Snort 구조는 PPM 실행부터 DM 실행이 끝날 때까지 패킷의 수집이 이루어지지 않는 특징을 가진다.

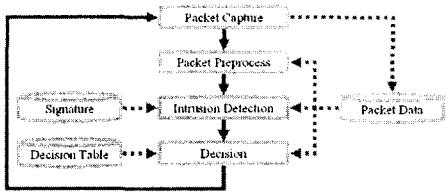

그림 7. Snort 구조

본 논문은 Snort 구조를 수정하여 그림 5와 같은 멀티 쓰레드 기반 N-IDS 모델인 Crasto를 설계하였다. Thread Manager(이하 TM)는 패킷의 캡쳐와 탐지를 관리한다. Detection Engine(이하 DE)은 여러 개가 생성된다. 각 DE는 미리 지정된 Packet Data(이하 PD)의 특정 영역을 할당받는다. Decision Manager는 Decision Table을 이용하여 DE로부터 탐지된 보안 침해에 대응한다.

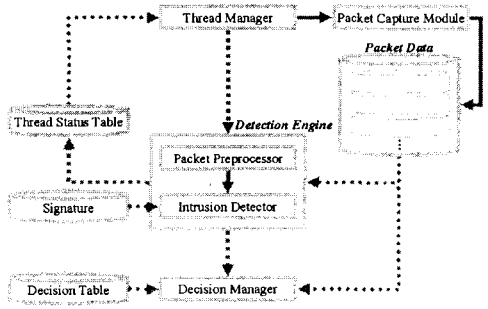

그림 8. 멀티 쓰레드 기반 N-IDS 모델

TM이 PCM을 호출하면 PCM은 data link layer로부터 직접 패킷을 읽어 들여 TM이 지정한 PD 내부의 특정 위치에 저장한다. PD 내부의 특정 위치는 TM이 PCM을 호출하기 전에 Thread Status Table(이하 TST)을 참조하여 미리 선택한다. TM은 PCM의 동작이 끝나면 PD 내부의 특정 위치를 사용하는 DE에게 탐지를 지시한 후 wait 상태인 쓰레드를 TST에서 찾아 탐지 작업을 반복한다.

3.2 쓰레드간 통신 구조

Crasto는 TM과 DE가 쓰레드로 생성되는 구조를 가지고 있다. 생성된 쓰레드를 관리하기 위해

쓰레드간의 통신이 필요하다. 이를 지원하기 위해 BSD file socket을 사용하였다. 기존의 IPC(Inter Processor Communication)나 signal 함수를 사용할 수도 있으나 후에 확장이 용이하도록 socket을 사용하였다. 생성되는 두 개의 manager는 다수의 DE와 통신을 위해 각각의 경로가 필요하게 되므로, 두 개의 file socket의 생성이 필요하다. 그림 6은 file socket structure에서의 데이터 흐름을 나타낸다.

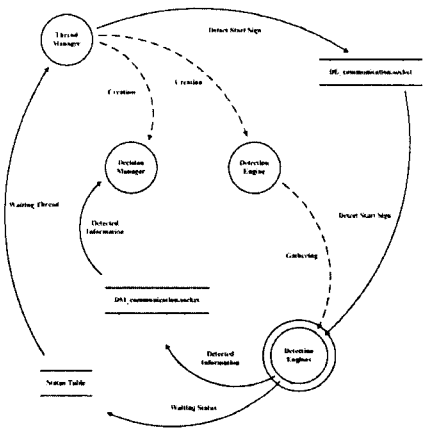

그림 9. File Socket Structure의 데이터 흐름도

4. 구현 및 성능 평가

4.1 test bed

본 논문의 test bed는 그림 7과 같다.

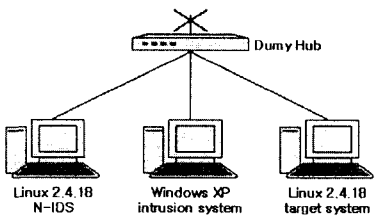

그림 10. test bed 구성도

4.2 성능 평가

평가의 정확성을 위해 동일한 침입 패킷 패턴과 탐지 모듈, 룰을 사용하여 성능을 평가하였으며 크게 두 가지 형태로 실시하였다. 하나는 쓰레드를 생성하지 않는 기존의 구조를 가지는 시스템과 멀티 쓰레드 구조를 가지는 시스템의 지연 시간을 측정해 보았고, 다른 하나는 구현된 시스템에서 생성되는 쓰레드 개수가 지연 시간에 미치는 영향을 대해서 측정해 보았다.

4.2.1 단일 프로세서 vs. 멀티 쓰레드

먼저 단일 프로세서를 생성한 시스템과 10개의 쓰레드를 생성한 시스템에서 시간 지연을 비교해 보았다. 테스트의 정확성을 위해 129개의 침입 패턴을 세 번으로 나누어 43개 단위로 성능 평가를 실시하였다.

4.2.1.1 사례 1

그림 8에서 단일 프로세서 방식은 불규칙한 그래프를 생성하고 있다. 이것은 패킷의 signature에 따라 처리 시간을 달리하기 때문이라고 추측해 볼 수 있다 이에 비해 멀티 쓰레드 방식은 규칙적인 그래프를 생성한다. 각 방식의 평균 지연 시간은 단일 프로세서 방식에서는 0.573ms, 멀티 쓰레드 방식에서는 0.091ms이다.

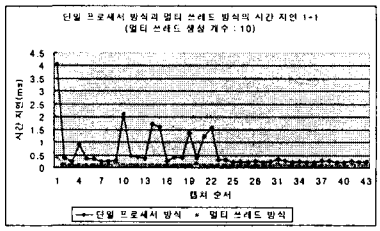

그림 11. 단일 프로세서 vs. 멀티 쓰레드 #1

4.2.1.2 사례 2

그림 9에서 각 방식의 평균 지연 시간은 단일 프로세서 방식 4.154ms, 멀티 쓰레드 방식 0.268ms이다.

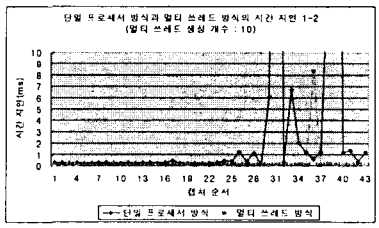

그림 12. 단일 프로세서 vs. 멀티 쓰레드 #2

4.2.1.3 사례 3

그림 10에서 각 방식의 평균 지연 시간은 단일 프로세서 방식 15.655ms, 멀티 쓰레드 방식 3.382ms이다.

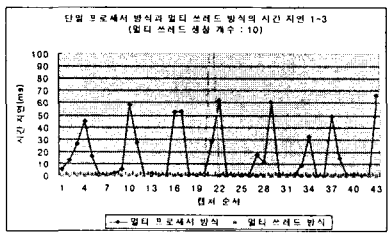

그림 13. 단일 프로세서 vs. 멀티 쓰레드 #3

지금까지 단일 프로세서 방식과 멀티 쓰레드 방식에서 일어나는 지연 시간을 비교해 보았다. 3가지의 사례에서 본 논문이 구현한 시스템은 기존 구조의 시스템에 비해 최소 5배 이상의 성능 향상을 확인할 수 있었다.

4.2.2 쓰레드 생성 개수에 따른 지연 시간 측정

다음으로 쓰레드 생성 개수에 따른 시간 지연을 측정하였다. 정확성을 높이기 위해 패킷, 패킷의 생성 개수, 패킷의 생성 시간을 모두 동일하게 설정하였다. 이를 위하여 사용된 패킷 생성기와 패킷은 아래와 같다.

- 패킷 생성기
- Visual Traffic Generator[11]
- 패킷 정보
- protocol type : tcp
- intrusion type : tcp null scan
- source & destination port : 5555

그림 11은 Crasto에서 생성되는 쓰레드의 개수를 변화시키면서 tcp null scan 패킷을 탐지할 때의 지연 시간을 측정한 것이다. 생성된 패킷의 개수는 총 40개이며 Visual Traffic Generator[11]에 의해 동일한 속도로 네트워크에 전송되었다.

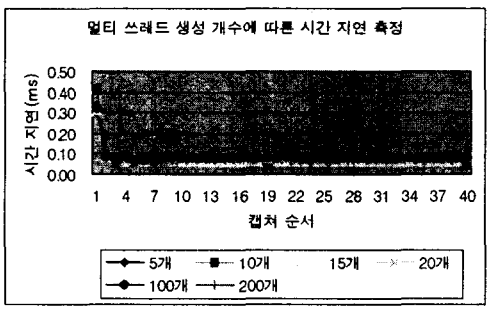

그림 14. 멀티 쓰레드 생성 개수에 따른 시간 지연 측정

Crasto에서 5, 10, 15, 20, 100, 200개의 쓰레드를 생성하여 측정한 결과 캡쳐 시간 지연의 평균 속도는 각각 0.068ms, 0.084ms, 0.067ms, 0.077ms, 0.081ms, 0.081ms, 0.082ms 이었다. 15개의 쓰레드를 생성하였을 경우 캡쳐 시간 지연의 평균 속도가 가장 낮게 나타났고, 그림 11에서 알 수 있듯이 그래프가 고른 모양을 나타내었다. 20개 이상의 쓰레드 생성은 15개의 쓰레드를 생성한 경우와 비교해 보면 캡쳐 지연 시간이 더 증가하므로 20개 이상의 쓰레드 생성은 Crasto의 성능 향상에 의미가 없다.

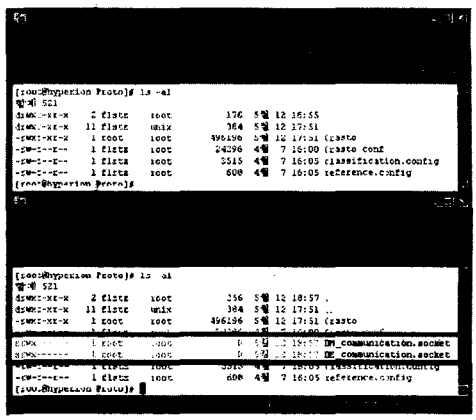

그림 15. 구현된 시스템에 의해 생성된 socket file

4.3 socket file의 생성

본 논문에서는 socket file을 생성하여 쓰레드간의 통신 및 제어가 이루어진다. 그림 12는 본 논문이 구현한 시스템을 실행하였을 경우 생성된 socket file의 정보를 표시하고 있다.

5. 결론 및 향후 연구 과제

본 논문은 기존 N-IDS의 구조적 문제점을 개선하기 위해 멀티 쓰레드 구조를 가지는 Crasto를 설계 및 구현하였다. Crasto는 기능 단위 구조가 아닌 쓰레드 단위 구조와 file socket을 이용한 통신 구조를 가지며, 전처리와 탐지 기능이 통합되어 한 쓰레드에서 동작하도록 설계 및 구현하였다. Crasto를 평가한 결과 기존 구조에 비해 캡쳐 지연 시간이 최소 5배 이상 단축되었으며, 지연 시간이 일정하게 유지되는 것을 관찰할 수 있었다. 본 논문의 향후 연구 과제로는 설치된 HW의 성능이나 네트워크 부하 따라 지능적이고 동적으로 쓰레드를 관리할 수 있는 부분의 연구가 필요하다.

참고 문헌

[1] 임채훈 외, 인터넷 보안 : 가상사설망, 방화벽 그리고 침입탐지시스템, The Proceedings Of The Korean Institute Of Communication Sciences 2000, 한국통신학회지, pp.97-110, Vol. 17, No. 3, Mar. 2000

[2] 임채훈, "인터넷 보안기술 개론-침입 탐지 시스템", Technical Report FS-TR00-10, 퓨처시스템 암호체계센터, Oct. 2000

[3] 이주남 외, "침입 탐지 시스템 구현을 통한 문제점 및 개선 방안에 관한 연구", 정보과학회 춘계 학술발표논문집, Vol. 28, No. 1, Apl. 2001

[4] Koral Ilgun. "USTAT - A Real-time Intr-

usion Detection System for UNIX", Technical Report, University of California at Santa Barbara, Nov. 1992
[5] Herve Debar, Marc Dacier and Andres Wespi, "Towards a Taxonomy of Intrusion Detection Systems", Research Report of IBM Research Division, Zurich Research Laboratory, Jun. 1998
[6] 정훈조 외, "침입의 유형과 탐지 시스템의 분류", 한국정보처리학회 추계 학술발표논문집, Vol. 6, No. 2, Oct. 1999
[7] Stephen Northcutt, Judy Novak, Network Intrusion Detection : An Anlyst's Handbook Third Edition, New Riders, Aug. 2002
[8] Teresa F Lunt, "A survey of intrusion detection techniques", In Computers and Security, pp.405-418, Dec. 1993
[9] Martin Roesch, Snort - Lightweight Intrusion Detection for Networks, In Proceedings of LISA'99:13th Systems Administration Conference, pp.229-238, Nov. 1999
[10] pcap library, http://www.tcpdump.org
[11] 정인환 외, "비쥬얼 이더넷 트래픽 발생기의 설계 및 구현", 한국 정보처리학회 추계 학술발표논문집, Vol 9, No.2, Oct. 2002