제20회 한국정보처리학회 추계학술발표대회 논문집 제10권 제2호 (2003. 11)

고속 네트워크 환경을 위한 패킷 필터링 시스템 설계

류승호*, 김정녀*
*한국전자통신연구원(ETRI)
e-mail : {ryush, jnkim}@etri.re.kr

Design of Packet Filtering System for High Speed Networking Environment

Seungho Ryu*, Jeong-Nyeo Kim *
* Electronics and Telecommunications Research Institute

요 약

본 논문에서는 고속 네트워킹 환경을 위한 패킷 필터링 시스템 설계 기법을 제안한다. 제안하는 기법은 기존 리눅스 운영체제에서 동작하는 패킷 필터링 구조의 단점을 개선하기 위하여, 패킷 필터링 규칙 저장 시 특정 커널 메모리 영역을 할당하여 패킷 검사와 관련된 모든 규칙을 취합하여 저장하고, 패킷 검사 시에 할당된 메모리 영역에서 규칙을 한꺼번에 접근하여 검사하는 방법이다. 또한 규칙의 크기를 고정화하여 규칙 검색 시 규칙 저장 위치를 간단하게 계산할 수 있도록 하였다. 이로 인해 기존의 테이블 구조에서 지니고 있던 다단계 테이블 검색으로 인한 메모리 참조 시간을 줄이고, 가변 규칙으로 인한 계산의 번거로움을 해소할 수 있다. 이를 통하여 고속 네트워크 노드 환경에서의 패킷 필터링 기능을 효율적으로 지원할 수 있다.

Keywords: Memory Page, Packet Filtering, Linux, Operating System, Network Security

1. 서론

리눅스 환경에서 패킷 필터링 기능은 Netfilter[1]를 이용하여 지원된다. Netfilter 는 리눅스의 TCP/IP 스택에서 미리 일정한 지점에 패킷을 조작하는 함수를 등록할 수 있도록 한 구조를 말한다. 패킷 필터링은 이를 이용하여 패킷이 나가는 곳과 들어오는 곳에 hooking 함수를 두고, 이 함수는 미리 정해둔 규칙을 기준으로 함수로 넘어온 패킷을 검사하여 처리하게 된다.

문제는 리눅스에 사용하는 패킷 필터링 규칙의 저장 방식이 다단계 접근 방식이기 때문에 고속 네트워킹 환경 하에서 패킷 입출력이 빈번한 시스템의 경우 패킷 필터링 기능이 병목 지점으로 작용할 가능성이 크다는 것이다. 따라서 패킷 필터링 규칙 검색을 단순화할 수 있는 기법이 필요하다.

2. 문제점

본 절에서는 리눅스에서 사용하고 있는 패킷 필터링 규칙 저장 방법에 대하여 설명한다.

2.1 규칙 테이블 자료 구조

리눅스의 패킷 필터링 기법은 테이블 중심구조를 취하고 있다. 먼저 패킷 필터링 규칙을 저장하고 있는 테이블에 접근한 후, 테이블에 기록된 규칙의 위치 정보를 확인하고, 테이블에 기록된 정보를 바탕으로 규칙의 위치를 계산하여 규칙에 접근하도록 되어 있다. 문제는 다양한 기능을 위해 여러 개의 테이블을 만들 경우에는 한 패킷을 검사하기 위해서 여러 테이블을 검색해야 하는 문제점이 발생한다. 예를 들어 현재 리눅스에서는 패킷 필터링을 위한 'filter' 테이블과 패킷 조작을 위한 'mangle' 테이블을 두어 각각의 기능을 지원하고 있는데, 만약 두개의 테이블이 모두 하나의 hooking 지점에 관련된 규칙을 가지고 있는 경우 매번 패킷을 검사할 때마다 두개의 테이블을 차례대로 검색해야 한다.

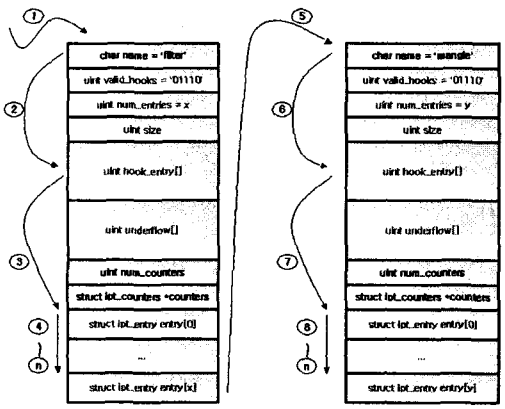

그림 1 리눅스 규칙 테이블 검색 과정

2.2 규칙 자료 구조

리눅스에서 사용하는 패킷 필터링 규칙은 확장성을 위해 가변 길이 형식을 가지고 있다. 이런 규칙을 테이블 하단부에 차례대로 저장하고 있기 때문에 여러 규칙을 검색하기 위해서는 앞에 있던 규칙에서 크기를 더하는 방식으로 순차적으로 계산해 나가야 한다. 이 방법은 확장성이라는 측면에서는 장점이 있지만, 규칙 검색 시간이 많이 소모되고, 매 규칙마다 크기를 저장하는 필드들이 곳곳에 배치되어 있기 때문에 결과적으로 메모리의 낭비를 가져오게 된다.

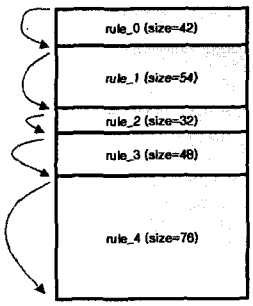

그림 2 가변 규칙 검색 단계

3. 규칙 테이블 변경 방법

본 절에서는 고속 규칙 검색을 위한 적용 기법을 설명한다.

3.1 규칙 테이블 구조

테이블 중심 구조인 리눅스 패킷 필터링 방법을 hooking 중심 자료 구조로 변환한다. Hooking 중심 자료 구조는 규칙을 커널 메모리에 저장할 때, 미리 hooking 지점별로 규칙을 저장할 메모리 영역을 확보하고, 모든 테이블의 규칙을 가져와서 우선 순위에 맞게 배열하여 패킷이 도착하면 여러 테이블을 검색하지 않고 배열된 순서에 따라 검사하도록 하는 방법이다. 이 방법은 기존의 리눅스의 규칙 테이블 구조를 최소한의 변경만으로 가능하도록 함으로써 복잡한 작업을 필요로 하지 않는다.

그림 3 에 변경된 규칙 저장 방법이 있다. 여러 테이블에 분산된 규칙을 hooking 지점별로 할당된 메모리 영역에 모아 두고 각 테이블은 해당 메모리 영역에서의 상대적인 위치를 저장한다. 여러 테이블에서 규칙을 모아 분류하는 경우 우선 순위에 맞게 규칙을 분류하여 저장해야 한다.

이렇게 하면 (1) 규칙의 변경 시에 매번 규칙을 모아 순서를 계산하여 배치하여야 하기 때문에 시간이 많이 걸리고 (2) 사용하는 규칙의 개수가 적을 경우 미리 할당해 둔 메모리가 낭비되는 단점이 있으나, (1) 규칙의 변경 작업은 관리자의 보안 정책을 수정할 때나 긴급 상황에서 발생하는 것이 통상적인 경우로서 자주 일어나지 않으며, 고속 환경에서의 패킷 검사 주기(㎳, ㎲)와 규칙 변경 주기(초, 분, 시간, 일, 주 등)는 시간의 단위 비교를 논할 필요가 없을 만큼 차이가 크다. (2) 패킷 검사 시에는 해당 메모리 영역의 규칙만을 차례로 검사하면 되기 때문에 다단계 메모리 참조로 인한 시간을 줄일 수 있고, (3) 여러 테이블의 경우는 물론 하나의 테이블 만을 대상으로 할 경우에서도 규칙이 저장된 영역을 직접적으로 접근할 수 있어서 고속의 검색이 가능하다. (4) 또한, 낭비되는 메모리의 경우 통계적인 수치 획득(예. 평균 사용 규칙 개수, 최대 및 최소 범위 사용 규칙 개수 등)으로 초기 할당 메모리의 크기를 조절(tuning)하면, 낭비되는 메모리의 양을 줄일 수 있다.

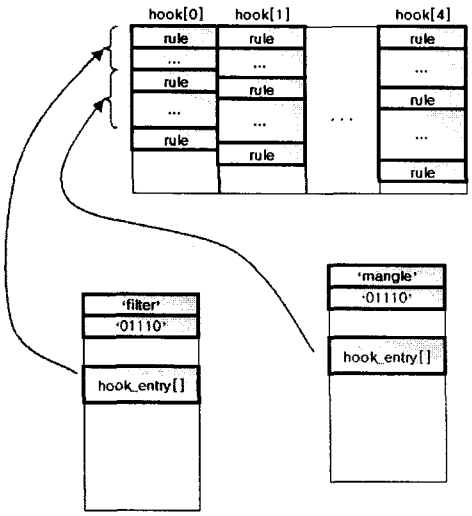

그림 3 변경된 규칙 저장 방법

3.2 규칙 고정화

규칙의 크기를 미리 동일하게 고정해 놓는다. 사용하려는 규칙의 종류를 미리 분류하고, 이 규칙만을 모두 포괄할 수 있는 최소 크기의 규칙을 정해서 저장한다. 이렇게 하면 순차적으로 규칙의 크기를 계산할 필요 없이 고정 크기의 배수로 규칙을 저장하므로 간단한 연산으로 위치를 찾아낼 수 있다.

이 방법은 (1) 차후에 새로운 형태의 규칙을 적용하려고 할 때, 만약 적용하려는 규칙의 크기가 미리 고정시켜둔 규칙의 최대 크기를 넘게 된다면 규칙 적용이 불가능하다는 단점이 있으나 (1) 새로운 형태의 필터링 규칙을 만드는 경우가 흔하지 않고, (2) 만약 새로운 형태의 규칙을 지원하고 싶은 경우에는 그 규칙을 고려한 규칙 자료 구조의 최대 크기를 재조정하고 설정하는 과정을 통하여 지원이 가능하다. (3) 또한 크기를 저장하는 영역이 필요가 없기 때문에 부수적으로 구조체의 크기를 줄일 수 있는 장점이 있다.

4. 결론

본 논문에서는 고속 네트워킹 환경을 위한 패킷 필터링 시스템 설계 방법을 제안하였다. 제안한 기법은 각 테이블에 걸친 여러 규칙 정보를

미리 할당된 메모리 영역에 한데 모아 저장하고, 각 규칙도 크기를 통일하여 배치하는 hooking 중심 자료 구조 방법이다. 이 방법은 메모리 접근 단계를 줄임으로서 빠른 시간에 규칙의 검색이 가능하고, 규칙이 통일화 되어 있어서 단순한 연산만으로 규칙의 위치를 찾아낼 수 있다는 장점이 있다. 따라서 이 기법을 통하여 고속 네트워크 환경에서 패킷 필터링 기능을 효과적으로 지원할 수 있다. 향후 연구 과제로는 성능 측정 및 비교 작업, 최적의 규칙 메모리 할당 방법과 보다 세밀한 규칙 항목의 최적화에 대한 연구가 필요하다.

참고문헌

[1] Netfilter/Iptables, http://www.netfilter.org

[2] Linux 2.4 Packet Filtering HOWTO, Rusty Russell, http://www.netfilter.org/documentation/HOWTO/packet-filtering-HOWTO.html

[3] Iptables Tutorial, http://iptables-tutorial.frozentux.net/iptables-tutorial.html