홈네트워크 환경에서 커널모듈을 이용한 유해사이트 차단[1]

박인성[U] 김홍철 송병욱 김상욱
경 북 대 학 교 컴 퓨 터 과 학 과
{ispark, hkkim, bwsong, swkim}@woorisol.knu.ac.kr

Harmful Web Site Blocking Using Kernel Module In Home Network Enviornment

In-sung Park[U] Hong-Chul Kim Byung-Wook Song Sang-Wook Kim
Dept. of Computer Science, Kyungpook University

요 약

최근 정보가전과 홈네트워크산업이 급속히 발달됨에 따라 가정이 하나의 네트워크망을 형성하고 있다. 이에 홈네트워크에서도 보안의 필요성을 인식하게 되었으며, 유해사이트 차단은 필수가 되었다. 이러한 사이트 차단을 위해 기존의 유닉스, 리눅스는 libpcap 라이브러리를 윈도우즈는 NDIS의 API를 이용한 응용계층에서의 패킷 필터링방법을 지원한다. 하지만 이와 같은 방법은 가정의 시스템 성능과 지속적으로 추가되는 홈서버의 기능으로 인해 시스템에 많은 부하를 발생시킨다. 본 논문은 이러한 부하를 최소화하기 위해 시스템 성능이 좋은 커널 모듈을 이용한 패킷 필터링 방법을 제시하고 이를 활용한 유해사이트 차단 시스템의 구현 예를 보인다.

1. 서론

홈네트워크 방화벽은 적용대상이 일반 가정이기 때문에 유해사이트 차단기능은 필수적이라 할 수 있다. 현재 지키미[1], 수호천사[2]등 많은 제품들이 판매되고 있다. 하지만 많은 유해사이트 차단 프로그램은 대부분 윈도우즈(Windows)기반으로 각 클라이언트에 대해서 차단기능을 제공하고 일부제품만이 네트워크 전체를 차단한다. 홈네트워크에는 네트워크 사용자 전체를 제어할 수 있는 제품이 효율적이지만, 현재의 제품들은 유해사이트 차단을 위해 서버에 너무 많은 성능을 요구한다. 홈네트워크의 홈서버는 특성상 성능이 낮고, 가전기기나 가정내 여러 시스템을 보호하고 서비스하는 많은 작업들을 수행해야 하기 때문에, 유해사이트 차단시스템은 시스템 자원을 최대한 적게 사용하도록 설계되어야 한다. 따라서 본 논문에서는 낮은 성능의 홈서버가 가정내 정보나 기기들에 대한 보호 서비스도 동시에 수행할 수 있도록 부하가 적은 효율적인 유해사이트 차단 모델을 제시 제시한다.

2절에서는 일반적인 유해사이트 차단유형과 실제 차단에 사용되는 필터링 기법의 문제점을 알아보고 3절에서는 이러한 문제점 해결을 위한 새로운 필터링 모델을 제안한다. 4절에서는 시스템의 구현예를 보이고 마지막으로 결론을 내린다.

1. 본 연구는 정보통신연구진흥원이 지원하는 이동네트워크 정보보호기술개발 연구의 일부분임.

2. 일반적인 유해사이트 차단

유해사이트 차단 방법은 크게 클라이언트기반 방법과 서버기반 방법의 두가지로 분류된다. 현재까지는 클라이언트기반 방법이 많이 이용되었으나, 최근에는 기간망 및 가입자망의 확산으로 인해 학교나 기업등에서 서버기반 방법의 이용이 급속히 증가하고 있다.

2.1 클라이언트기반 유해사이트 차단

클라이언트기반의 유해사이트 차단은 주로 유해사이트 목록, 차단프로그램으로 구성된다. 웹브라우저로부터 주소를 추출하는 기술과 추출한 주소를 클라이언트상의 유해사이트 목록과 비교하여 접속을 허가 또는 차단하는 기술이 필요하다. 사용자의 웹브라우저마다 구현해야 하고, 사용자가 여러명일 경우에 각각의 클라이언트마다 프로그램과 목록을 설치해야 한다.

2.2 서버기반 유해사이트 차단

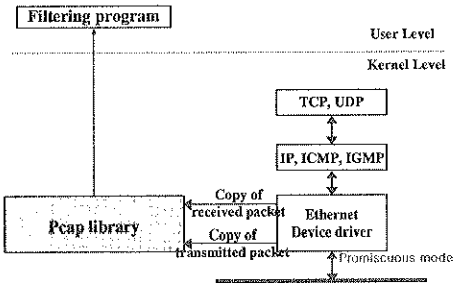
서버기반의 차단 프로그램은 홈네트워크내의 클라이언트 웹접속을 감시할 수 있는 필터링기술과 홈네트워크내 패킷의 외부네트워크 유출 통제 기술이 필요하다. 실제 작동방식은 홈네트워크내의 클라이언트들이 웹브

라우저를 통해 특정 웹사이트에 접속하고자 할 경우 홈서버의 차단 프로그램은 외부로 나가려는 패킷을 필터링하여 목적지 주소를 알아낸다. 이주소는 유해사이트목록과 비교되고 일치할 경우 그 패킷의 외부 유출을 막는 것이다. 이는 홈서버에만 차단프로그램을 설치하여 홈네트워크내의 모든 클라이언트들을 통제할 수 있어서 효율적이다.

2.3 일반적인 필터링 모델

홈네크워크의 유해사이트 차단 시스템은 홈서버의 방화벽내에 구현되는데 기본적으로 패킷을 허용 또는 차단하기 위해 패킷 모니터링을 할 수 있어야한다.

일반적으로 모니터링을 위한 패킷 캡쳐 방법으로 유닉스, 리눅스에서는 libpcap 라이브러리를 윈도우즈에서는 NDIS를 사용한다. libpcap 라이브러리는 네트워크카드를 Promiscuous mode로 바꾸어 네트워크 상의 모든 패킷을 받아 볼 수 있도록 한다.



(그림 1) libpcap을 이용한 패킷 필터링 모델

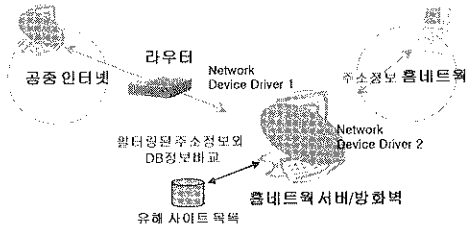
윈도우즈에서는 특별히 필터링하는 공개된 라이브러리가 존재하지 않고 NDIS(Network Driver Interface Specification)라는 인터페이스와 환경을제공한다. 네트워크 카드를 제조하는 회사는 윈도우즈 특유의 전송 드라이버를 작성하는 대신에, 단일 네트워크 드라이버의 최상층으로서 NDIS 인터페이스를 제공한다. 이렇게 함으로써 어떤 프로토콜 드라이버도 이 인터페이스를 호출하여 자신의 네트워크 요구를 네트워크 카드로 지시할 수 있다.[3]

3. 커널모듈 기반의 패킷필터링 모델

홈네트워크의 홈서버는 일반기업과 달리 상대적으로 시스템 성능이 낮고, 홈네트워크내의 전자 기기등 많은 자원들을 통제, 보호하고 사용자에게 서비스 한다. 그러므로 유해사이트 차단 시스템은 홈서버의 시스템 성능과 앞으로도 계속적으로 추가될 여러 기능들을 고려하여 설계되어야 한다. 그러므로 유해사이트 차단을 위한 패킷모니터링은 시스템의 부하를 최소화하여 다른 프로세스 실행에 지장이 없도록 하여야 한다.

3.1. 유해사이트 차단 시스템망 구성

(그림2)는 유해사이트 차단을 위한 기본모델이다. 홈네트워크내의 클라이언트가 인터넷의 특정사이트로 접속하려하면 홈서버의 Network Device Driver2로 해당 패킷을 보내야 한다. Network Device Driver2에 수신된 패킷은 특정 프로세스가 캡처하여 유해사이트목록과 매치시키며 매칭결과에 따라 Network Device Driver1을 통한 패킷의 흐름을 허용하거나 차단한다. 즉 홈서버와 외부 인터넷망으로의 통신은 Network Device Driver1을 통해서 이루어지며 홈네트워크내 클라이언트들과는 Network Device Driver2를 통해서만 통신한다.[6]



(그림 2) 홈네트워크 방화벽의 유해사이트 차단

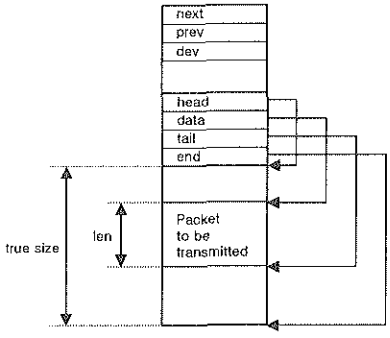
3.2 커널모듈 기반 패킷필터링

패킷을 필터링하는 최적의 방법은 패킷필터링 기능을 커널에 직접 구현하는 것이다. 이는 모니터링된 정보를 사용자레벨의 프로세스에게 전달하고, 그 프로세스가 전달된 패킷정보를 분석하여 처리명령을 반환하는 과정을 생략하므로 매우 효율적이다. 하지만 이것은 성능에 있어서일 뿐 기능에 변경이 있을 때마다 커널이 재컴파일되어야 하기 때문에 유연성은 만족하지 못한다. 이러한 문제점을 극복하고 성능과 유연성을 만족하는 방법은 바로 커널모듈을 이용하는 것이다. 리눅스는 단일(monolithic)커널로, 모든 기능적인 요소들이 자신의 내부 자료구조와 함수들에 접근할 수 있는 단일 프로그램이다. 또한 필요한 기능을 모듈로 구현하여 동적으로 커널에 로드 또는 언로드 할 수 있도록 커널모듈을 지원한다. 이러한 커널모듈은 커널의 일부로 작동하며, 커널 모듈 기능 변경시 단일 모듈만 수정하면 되므로 성능과 유연성을 동시에 갖게 된다. 리눅스는 프로토콜 계층 사이와 네트워크 디바이스 드라이버간에 데이터를 전달하기 위해 sk_buff라는 소켓 버퍼를 사용한다.(그림3)

각 sk_buff는 자신과 연관된 데이터 블록을 가지고 있다. sk_buff는 네 개의 데이터 포인터를 가지고 있는데, 이들은 소켓 버퍼 데이터를 다루고 관리하는데 사용된다. 네트워크 장치는 네트워크로부터 패킷을 수신하면 이 수신한 데이터를 sk_buff자료구조로 바꾼다. 네트워크 드라이버는 이들을 수신할 때마다 backlog queue에 수신한 sk_buff들을 추가하는데, 이때 스케줄러는 네트

워크 하반부(bottom half) 핸들러를 실행하고 이는 sk_buff 의 backlog queue를 처리하기 이전에 수신한 패킷을 어떤 프로토콜 계층으로 전달할지를 결정한다.

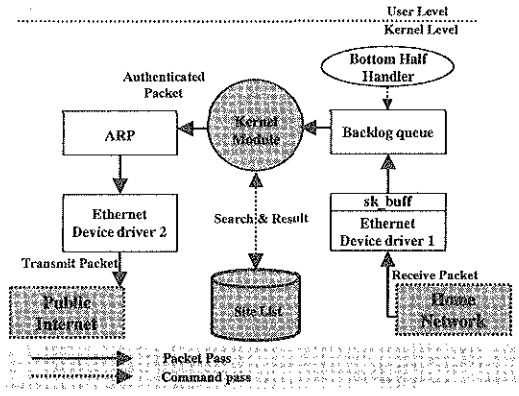
리눅스는 네트워킹계층을 초기화할 때 각 프로토콜은 packet_type 자료구조를 ptype_all 리스트나 ptype_base 해시테이블에 추가함으로서 자신들을 등록한다. 네트워크 하반부는 들어오는 sk_buff의 프로토콜 타입과 각 테이블에 있는 하나 이상의 packet_type 엔트리와 매치시킨다. 프로토콜은 하나 이상의 엔트리와 매치될 수 있는데, 이를 이용해 모든 네트워크 트래픽을 볼 수 있다.[4]



(그림 3) Socket Kernel Buffer(sk_buff)

4. 시스템과 구현 예

본 연구에서는 홈네트워크 유해사이트 차단 시스템의 성능과 유동성 극대화를 위해 (그림 4)와 같이 리눅스 커널모듈을 이용한 IP기반의 유해사이트 차단 시스템을 구현하였다. 차단시스템은 홈네트워크와 연결되어 들어오는 패킷을 sk_buff구조로 변환하는 Ethernet Device Driver1과 외부인터넷에 연결되어 외부로 패킷을 전송하는 Ethernet Device Driver2 그리고 이들 사이에서 시스템 내부적으로 패킷을 인증 여부에 따라 차단 또는 허용하는 커널모듈들로 구성된다. 커널모듈은 하반부핸들러(bottom half handler)가 backlog queue로부터 보낸 패킷의 IP헤더와 유해사이트 목록의 자료를 비교하고 인증여부를 결정하며, 인증된 패킷이 ARP를 거쳐 전송할 Ethernet Device Driver2를 통해 외부로 나가게 한다. 내부적으로 device driver에서 나와 sk_buff로 들어가는 모든 패킷을 조작하기 위해서는 packet_type{} 데이터 구조를 등록해야 하기 때문에 packet_type{}구조에 필터링할 프로토콜을 저장하고, dev_add_pack()함수를 사용하여 이를 등록했다. 핸들러는 디바이스 드라이버와 다음 순서의 핸들러 사이에 놓이게 되며, 디바이스 드라이버로 도착하는 모든 sk_buff는 제일 먼저 이 핸들러를 거쳐야만 한다. 그러므로 커널모듈은 핸들러를 통해 패킷을 모니터링하고 모니터링을 통해 패킷을 차단한다.



(그림 4) 커널모듈 유해사이트 차단 시스템 구조

5. 결론

본 논문에서는 홈네트워크 방화벽에 유해사이트 차단 시스템을 구현하기 위해 일반적인 패킷 필터링의 방법과 문제점을 알아보고, 이런 방법들의 문제점 개선책으로 커널모듈을 이용한 패킷 필터링 및 구현방법에 대해서 기술하였다. 이러한 결과로 구현된 커널모듈 패킷필터링 기반의 유해사이트 차단 시스템은 성능과 유동성에서 많은 장점을 가지게 되었다. 하지만 시스템 성능에만 너무 치중한 결과 실제 유해사이트를 차단하는 다양한 기법들에 대해서는 부족한 점이 많이 있다. 지속적으로 늘어가는 유해사이트의 목록을 사용자가 직접 입력해야하고, 홈네트워크내 사용자들에게 동일한 차단 정책이 적용된다. 향후 특정사이트나 파일로부터 유해사이트 목록을 자동 갱신하는 기능과 클라이언트들 각각에 대해 정책을 세우고, 이 정책에 따라 사이트를 차단할 수 있는 기능을 추가할 예정이다. 또한 최근 웹상의 컨텐츠 정보를 찾아다니면서 정보를 축적하는 웹봇의 연구가 활발히 진행되고 있는데, 이러한 기술과의 접목을 통해 구현된다면 더욱 효율적인 유해사이트 차단 시스템이 될 것으로 기대된다.

6. 참고 문헌

[1]인터피아월드(주), http://www.jikimi.net
[2]플러스기술(주), http://www.plustech.co.kr
[3]한국정보보호센터, "실시간 LAN기반 패킷 모니터링 시스템 개발", 1998.12
[4]David A Rusling, "The Linux Kenel", 1999
[5]Ori Pomerantz,"Linux Kernel Module Programming Guide", 1999
[6]Simon Garfinkel, Gene Spafford,Practical Unix & Internet Security, 2nd Edition, O'Reilly, 1996
[7]http://phrack.infonexus.com/search.phtml?view&article=p55-12 : Building Into The Linux Network Layer