클라우드 데이터베이스를 이용한 커널 기반 고속 패킷필터링 임베디드 게이트웨이 및 방화벽 개발

박 대 승* · 김 수 민* · 유 한 섭* · 문 송 철**

목 차

요약

본 논문에서는 라우터의 라우팅 기능, L7스위치의 컨텐츠 기반 보안, IPS 침입 방지 시스템의 기능과 클라우드, 빅데이터 기술을 혼합한 새로운 형태의 패턴분석 패킷 필터링 방화벽 장비를 저가형 임베디드 장비에서 구현해 SOHO(소규모 창업) 및 소규모 회사 또는 일반 가정의 보안 수준을 개선하고 새로운 시장을 개척하는 것을 목표로 하며, 또한, 고성능 스위치 및 기존 보안장비 기능 대체를 목표로 해 개발하고, 저 사양 장치에서 고성능 시스템을 구현할 수 있는 새로운 차세대 알고리즘을 제시한다.

표제어: 클라우드 데이터베이스, 커널 기반, 고속 패킷 필터링, 게이트웨이, 방화벽

접수일(2015년 3월 9일), 수정일(1차: 2015년 3월 13일), 게재확정일(2015년 3월 13일)

* 남서울대학교 컴퓨터학과 학부생
** 교신저자, 남서울대학교 컴퓨터학과 교수, moon@nsu.ac.kr

1. 서론

스마트시대에 접어들면서 인터넷은 우리의 일상으로 자리잡았다. 언제 어디서나 인터넷에 접근할 수 있는 유비쿼터스 시대에 들어선 것이다. 이러한 환경은 우리에게 많은 장점을 주지만 그 이면에는 단점 또한 존재하다. 우리는 이것을 보이지 않는 위협 즉, 보안침해라 말한다. 허가 되지 않은 정상적이지 않은 접근을 말하며 이익이 되는 사용자의 정보를 빼내거나 특정 대상에게 직접적인 피해를 주기 위한 네트워크 공격들이 있다. 하지만 이런 보안침해를 막기 위해서는 3세대 보안과 IPS(침입 방지 시스템)을 도입해야만 보안침해를 막을 수 있다. 그러나, 수천만 원까지 하는 이 고급 네트워크 장비들을 일반인들이 도입하기란 불가능하다. 따라서 본 논문에서는 저가형 임베디드 장비를 이용해 커널영역에서 동작하는 고성능 패턴분석 패킷필터링 방화벽을 개발하고, 클라우드 시스템을 이용해 정보를 수집, 통제 그리고 빅데이터 분석을 통한 예측 기능을 포함하는 장치를 개발한다. 이는 L7스위치의 컨텐츠 기반의 보안기능과 각종 네트워크 공격에 대해 적극적으로 방어하는 기능을 포함하는 침입 방지 시스템 기능을 포함해 일반 가정에서는 물론이고 현대 시대에 떠오르고 있는 스타트업(벤처)을 위한 정보보안 장비로 유일한 대안이 될 것이며, 새로운 시장을 개척하는데 도움을 주고 클라우드, 빅데이터, IoT (Internet of Thigns)에 대한 학문적 연구에 도움을 줄 것이다.

2. 관련 연구

2.1 보안

컴퓨터의 기능도 다양화되고 고도화되면서 범죄나 온라인 사고가 격증함에 따라 컴퓨터의 이용 기술, 데이터에 대한 피해나 재해를 방지하기 위한 보안 조치를 말하는데, 컴퓨터의 쓰임이 다양하고 광범위한 만큼 그에 대한 보안 조치도 다양하고 광범위하다.

2.2 Linux

리누스 토발즈가 유닉스를 기반으로 개발한 공개용 오퍼레이팅 시스템으로 소스코드를 완전 무료로 공개하고 커널부분을 개선해 특히 TCP/IP를 강력하게 지원하는 등 네트워크에 강점을 지녀 많은 사용자들이 서버로 사용하고 있다

2.3 Kernel

컴퓨터 운영체계의 가장 중요한 핵심으로 운영체계의 다른 모든 부분에 여러 가지 기본적인 서비스를 제공한다. 일반적으로 커널에는 종료된 입출력연산 등 커널의 서비스를 경쟁적으로 요구하는 모든 요청들을 처리하는 인터럽트 처리기와 어떤 프로그램들이 어떤 순서로 커널의 처리시간을 공유할 것인지 결정하는 스케줄러 그리고 스케줄이 끝나면 실제로 각 프로세스들에게 컴퓨터의 사용권을 부여하는 수퍼바이저 등이 있으며, 또한 커널은 메모리나 저장장치 내에서 운영체계의 주소공간을 관리하고 이들은 모든 주변장치들과 커널을 사용하는 사용자들에게 고루 나누어주는 메모리 관리자가 있다.

2.4 Kernel Module

커널 모듈이란 필요에 따라 커널에 로드하거나 언로드 할 수 있는 특정한 기능을 수행하는 코드로서, 운영체제를 재시작 하지 않고 커널의 기능을 확장 혹은 제거하거나, 원하는 기능을 수행할 수 있도록 하는 목적을 갖고 만들어진 것이다. 또한, 특정한 장치를 제어하는 코드를 작성시 커널 내부에 포함하면 필요하지 않은 사용자도 커널에 포함되 설치되어

쓰지 않는 자원낭비가 발생하게 된다. 이를 커널 모듈로 만들었다면, 단지 모듈을 언로드 시키면 되고 이것을 목적으로 만들어 진 것이 커널모듈이다.

2.5 Netfilter Framework

리눅스 커널 2.4버전 이상부터 제공되는 리눅스의 패킷 필터링 프레임워크로 패킷을 필터링하고 관리하며 Redesign할 수 있는 도구를 제공한다. Netfilter 프레임워크에 Hook을 이용해 기능을 이용할 수 있으며 Hook을 하면 수신된 패킷을 해당 프로그램으로 Callback을 해주게 되어 분석기능을 수행할 수 있게 도와준다.

2.6 Cloud Database

클라우드는 인터넷 기반의 컴퓨팅 기술을 말한다. 개인이 가진 단말기에서는 입/출력의 작업만 이루어지고, 정보분석 및 처리, 저장, 관리, 유통 등의 작업은 클라우드라고 불리는 제3의 공간에서 이루어지는 컴퓨팅 시스템의 한 형태이다. 이때, 데이터베이스만을 클라우드화 한 것으로 본 논문에서는 데이터베이스만 클라우드화 해 사용자들에게 최신 보안 위협을 업데이트 해주는 용도로 사용된다.

2.7 Google App Engine

클라우드 시스템의 가장 대표적인 Google App Engine(이하, GAE)은 Platform as a Service의 대표적인 서비스로 파이썬, 자바, PHP를 통해 Cloud Database 및 빅데이터 처리를 지원하는 것으로 스타트업 등의 자금력과 인력이 부족한 벤처기업 등에서 사용하는 것으로, 시스템 구축비용 없이, 단순히 사용료만을 지불하면 되기 때문에 각광받고 있는 서비스 중 하나이다.

3. 시스템 구성도

3.1 프로젝트 작업 구성

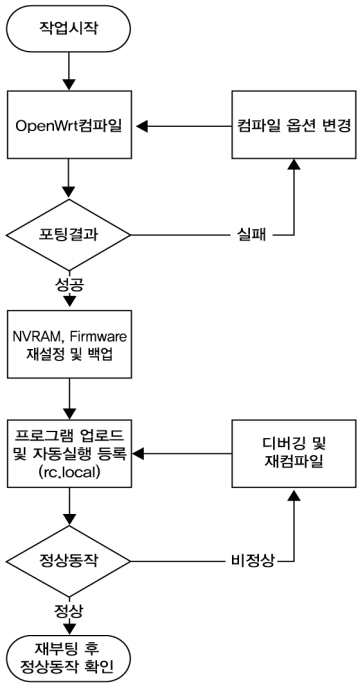

그림 1. 프로젝트 작업 구성
Fig. 1. Configuration of Project Work

3.2 프로그램 작업 구성

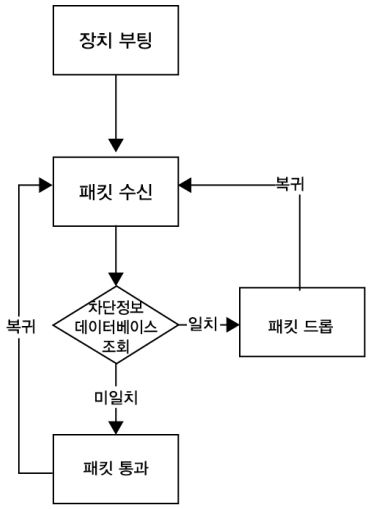

그림 2. 프로그램 작업 구성
Fig. 2. Configuration of Program Work

4. 기능

4.1 개요

Kernal에서 Ethernet 영역 내 TCP/IP 2계층(OSI 3계층)에서 동작하는 Netfilter Framework를 이용한 패킷 필터링을 구현하는 것을 목표로 한다. Netfilter Framework와 네트워크의 구조도는 제 5장에서 설명한다.

4.2 기능 요구사항

본 논문의 프로그램은 필수적으로 다음 기능을 구현한다.

표 1. 프로그램 기능 구현

Tab. 1. Construction of Program Function

장치	기능
L7 Switch	컨텐츠 기반 보안 기능
IPS(침입 방지 시스템)	침입 방지 및 능동적 대응 기능
Cloud Database	최신 보안위협에 대응

위에서 소개된 각 기능이 포함되어야 하는 이유는 다음과 같다.

4.2.1 L7 Switch

L7 Switch의 컨텐츠 기반 보안기능은 이동하는 트래픽 내부의 웹 기반의 데이터 내용을 보고 보안을 실현하는 것을 말한다. 본 논문에서는 트래픽의 패턴을 파악해야 함으로 컨텐츠 내용을 읽는 것은 필수이다. 하지만 L7 Switch에서는 일반적으로 HTTP헤더를 보고 해당하는 패킷의 문제 여부를 파악하게 되는데, 본 논문에서는 헤더뿐만이 아닌 Netfilter Framework의 장점을 이용해 헤더더 뿐만 아니라 패킷의 내용까지 읽어 문제 여부를 파악 더욱 정밀하고 정확하게 파악한다.

4.2.2 IPS(침입 방지 시스템)

IPS는 가장 중요한 요소로, 외부의 침입을 확인하였을 때 침입을 차단하며 장치가 능동적으로 대응하는 기능을 말하는데, 2000년대 초반까지도 IPS라는 개념이 존재하지 않았다. 결국은 침입이 이루어지고 난 후에 침입 탐지 시스템이 알려주는 것이기 때문이다. 따라서 이러한 문제점을 해결하고자 등장한 것이 침입 방지 시스템인데, 침입을 확인하였을 때 네트워크 관리자의 직접 대응없이 장치가 능동적으로 트래픽을 차단하고 Drop하며, 실질적인 보안을 실현하는데 가장 중요한 요소 중 하나이다.

4.2.3 Cloud Database

본 논문처럼 일반 사용자가 장비를 사용하는 경우, 가장 중요한 것은 장비의 소프트웨어와 정보를 최신으로 유지해주는 펌웨어 업데이트가 가장 중요하다. 하지만 일반 사용자의 경우 직접 장치를 사용하는 컴퓨터에 연결해 펌웨어 업데이트를 진행하는 것은 상당히 어렵기 때문에 하지 않는 경우가 대다수이다. 보안 장비에서는 이점이 상당한 위험요소로 자리잡을 수 있는데, 이를 보안하고자 클라우드 서비스인 구글 앱엔진을 이용해 사용자에게 원격에서 소프트웨어와 데이터베이스를 최신으로 유지하게 도와주며, 이를 통해 보안 위험 요소를 줄일 수 있고, 최신 위협요소에 재빠른 대응을 하여, 보안성을 보다 향상시킬 수 있다.

4.3 카테고리 검색 알고리즘

4.3.1 개요

본 논문의 주요 과제는 어떻게 고속처리를 할 수 있는가이다. 이전까지 사용하던 단순한 검색 알고리즘만으로는 기존까지의 성능이 한계이기 때문이다. 따라서 카테고리를 이용한 검색 알고리즘을 고안했다.

4.3.2 설명

이 알고리즘은 기존까지 전례가 없던 알고리즘이다.

본 논문에서는 보안에 대하여 4가지의 카테고리를 제시하고자 한다. 따라서 패킷이 장비로 들어오면 해당 패킷이 어느 카테고리에 해당하는지 기준항목들을 토대로 판단을 한 뒤 해당하는 카테고리 내에서만 검색을 수행한다. 이때 검색을 수행할 때는 랜덤억세스에 가까운 속도를 낼 수 있는 해시테이블을 이용하게 된다. 따라서 일반 검색속도와 비교할 때 필요한 부분에 대해서만 검색을 수행하게 되기 때문에 최적의 조건에서 검색속도가 1/n(n = 카테고리항목수)로 줄어들게 된다.

4.3.3 분류기준

본 논문에서 제시하는 분류는 크게 4가지로 구성되는데, 네트워크패킷, 데이터 패킷, 바이러스, Static Rules로 나뉘며 네트워크패킷은 Ping of Death 등의 네트워크 기반 공격을, 데이터 패킷은 파일 전송시 바이러스나 공격코드를 포함한 데이터파일을, 바이러스는 웜바이러스, 트로이목마, 스파이웨어, 애드웨어, 그레이웨어 등의 원격명령 및 비정상 작업을, Static Rules는 사용자가 설정한 규칙들로 예를 들어, “8000번 포트의 모든 패킷 차단”과 같은 작업을 각각 나누어 맡게 된다.

4.3.4 판단기준

IPv4 Header Format

Offsets	Octet	0							1							2							3						
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version				IHL				DSCP						ECN		Total Length														
4	32	Identification																Flags			Fragment Offset											
8	64	Time To Live								Protocol								Header Checksum														
12	96	Source IP Address																														
16	128	Destination IP Address																														
20	160	Options (if IHL > 5)																														

그림 3. IPv4 헤더 이미지
Fig. 3. IPv4 Header Image

그림 3의 IPv4 헤더 이미지를 기준으로 설명한다.

(1) 네트워크 패킷

네트워크 패킷은 위 IPv4 헤더내 Protocol 영역에 8bit가 0이 아닌 수로 되어 있으며 각 네트워크 패킷의 형식은 다음 표 2와 같다.

표 2. 각 네트워크 패킷의 형식
Tab. 2. Type of Each Network Packet

Protocol Number	Protocol Name	Abbreviation
1	Internet Control Message Protocol	ICMP
2	Internet Group Management Protocol	IGMP
6	Transmission Control Protocol	TCP
17	User Datagram Protocol	UDP
41	IPv6 encapsulation	ENCAP
89	Open Shortest Path First	OSPF
132	Stream Control Transmission Protocol	SCTP

따라서 위처럼 프로토콜 넘버가 있다면 네트워크 패킷으로 분류해 검사한다.

(2) 데이터 패킷(ASCII 형태)

데이터 패킷은 바이러스가 포함된 실행파일을 대상으로 하는 검사가 목적이며 일반적인 패킷은 다음과 같다.

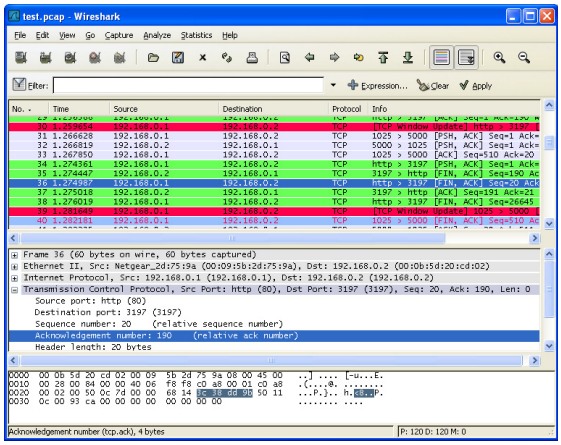

그림 4. 일반적인 패킷
Fig. 4. General Packet

일반적인 텍스트가 전송되는 파일은 IPv4 헤더 위에 Transmission Control Protocol에 탑재되어 전송된다.

(3) 데이터 패킷(바이너리 형태)

일반 ASCII 데이터 패킷과는 다르게 바이너리 데이터 패킷은 아래 이미지처럼 보여진다.

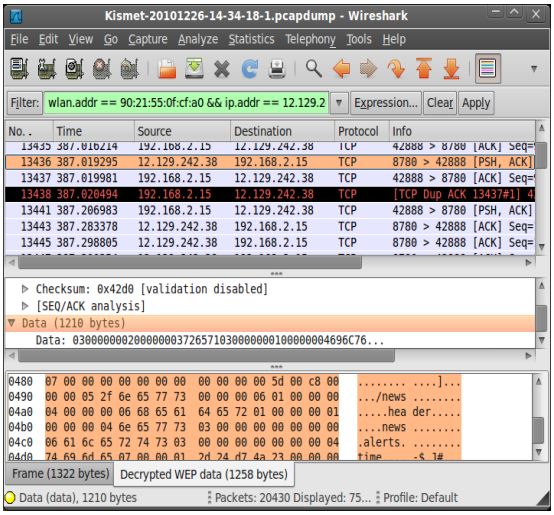

그림 5. 바이너리 데이터 패킷
Fig. 5. Bynary Data Packet

바이너리 데이터가 들어있는 헤더가 별도로 탑재되어 들어온다. 이것을 기준으로 해 바이너리값을 가진 패킷에 데이터 패킷 검사를 수행한다.

(4) Static Rules(iptable 기능 IP, port를 기준으로 한 규칙)

Static Rules는 사용자 편의 혹은 어드벤스 유저를 위해 제공되는 기능으로 iptable의 기본적 기능을 지원하며 IPv4 헤더 내 포함되어있는 Source IP, Port & Destination IP, Port에 대해 검사를 수행하고 규칙을 이행한다.

또한 Static Rules는 중요한 기능을 포함하고 있는데 IDS(침입 탐지 시스템)과 IPS(침입 방지 시스템)에서 침입을 탐지하고 지속적으로 발생하는 비정상 트래픽에 대해 능동적 대응을 하기 위하여 해당 Source IP를 반영구적으로 차단하기 위해서도 쓰여지게 된다.

4.4 네트워크 기능

4.4.1 QoS 기능

본 논문은 스위치를 대체하는 장치를 만드는 것이 목적이다. 따라서, 스위치의 기능을 무엇보다 원활하게 수행하는 것이 중요한데 스위치의 기능중 가장 중요한 것은 QoS(Quailty of Service) 즉, 서비스의 질이다. 얼마나 속도의 저하가 없이 서비스를 이용할 수 있는가에 대한 척도라 할 수 있는데 이 QoS는 최저속도 보장, 최고속도 제한, 속도 균등 분배 등을 통해 본 논문 장치에 연결되어 있는 Client들에게 특정한 Client가 네트워크 자원을 독점하지 않는 균등한 서비스의 질을 느낄 수 있도록 도와줄 필요가 있다. Linux Kernel 2.2 이상에서 지원하는 Traffic Control을 이용해 본 논문에서 QoS를 직접 구현하며, 현재 Target Device의 Kernel은 2.6으로 QoS를 구현하기에 매우 적합하다

4.4.2 Traffic Control 구조도

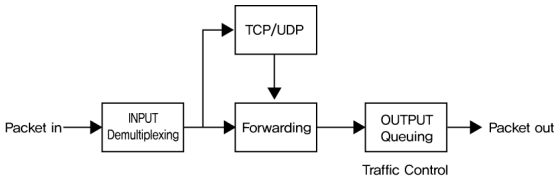

그림 6. 트래픽 컨트롤 구조도
Fig. 6. Traffic Control Structure

4.4.3 잠재적 위험 요소가 있는 패킷의 차단

본 논문의 최종적인 목표는 무엇보다 알려지지 않은 잠재적 위험 요소가 있는 패킷의 차단이며 이를 구현하려면 패턴을 인식하는 기능을 구현해야한다. 예를 들어, 텍스트 기반의 패킷 중 특정 위치 C:\Windows \와 같은 단어가 같은 SourceIP에서 지속적으로 들어올 경우 트로이목마 혹은 스파이웨어의 원격 명령일 가능성이 높으므로 사용자에게 알리거나 자동적으로 차단해줄 필요가 있다. 이러한 패턴을 다양하

게 읽고 학습해 장치 스스로가 사용자의 사용패턴에 맞게 변화하는 기능을 구현한다.

5. 구조

5.1 CPU & Linux Kernel Structure

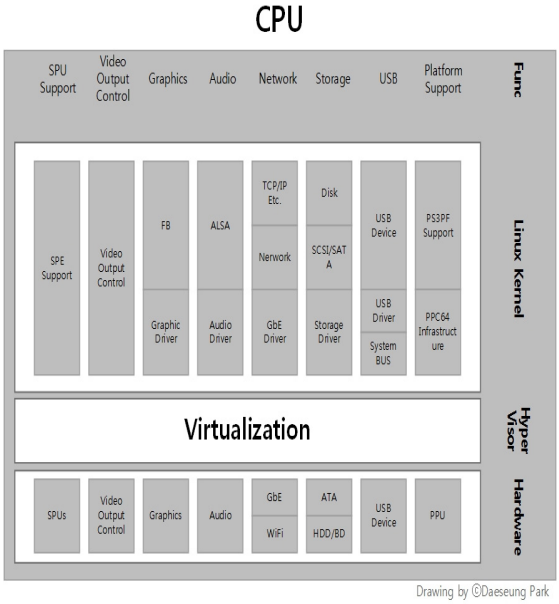

그림 7. CPU & Linux 커널 구조
Fig. 7. CPU & Linux Kernel Structure

5.2 TCP/IP Structure

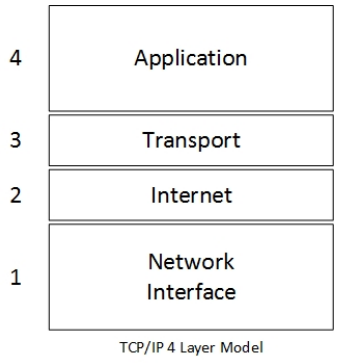

그림 8. TCP/IP 모델
Fig. 8. TCP/IP Model

5.3 Netfilter Structure

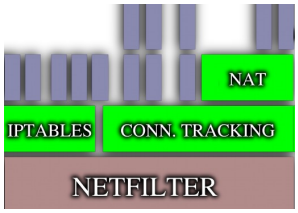

그림 9. Netfilter 구조
Fig. 9. Netfilter Structure

6. 시스템 구현

6.1 커널 모듈 구현[1)]

커널 모듈은 Linux C의 문법을 사용하지만 모든 부분에서 일반적인 C언어와는 다른 체계와 구조를 가지고 있다.

다음 3개의 헤더는 커널 모듈 구현 시 필수로 참조해야하는 헤더이다.

```
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>
```

다음 두 선언은 모듈의 배포자와 간단한 설명을 적어주는 것으로 커널 모듈 컴파일시 필수로 선언하도록 규정된 내용이다

```
#define DRIVER_AUTHOR "DRIVER_AUTHOR"
#define DRIVER_DESC   "Kernel Module"
```

다음 두 함수는 커널 모듈의 시작과 끝이다. 리눅스에서 해당 모듈을 인스톨 하면, init_modulestart 함수가 실행되고, 모듈을 언인스톨 하면, cleanup_modulefinish 함수가 실행된다. 주의할 점은 다음 과

1) 참고문헌(3) The Linux Documentation Project에 기술됨.

정에서 잘못되면 모듈이 어떠한 방법으로도 설치, 해제되지 않는 모듈잠금현상이 발생하는데, 이때는 재부팅으로 해결해야한다.

```
static int __init init_modulestart(void)
{
        printk(KERN_INFO "Hello, world 4\n");
        return 0;
}
static void __exit cleanup_modulefinish(void)
{
        printk(KERN_INFO "Goodbye, world 4\n");
}
```

다음 두 줄의 명령은 모듈의 시작과 끝 함수를 명시해주는 것으로, 일반적인 C언어는 main 함수가 시작이지만, 커널 모듈은 시작 함수 이름을 원하는 대로 만들 수 있기 때문에 명시해 주어야 한다.

```
module_init(init_modulestart);
module_exit(cleanup_modulefinish);
```

다음 명령은 커널 모듈에 대한 라이센스로 GPL, GPLv2 등이 있는데, 이는 커널 모듈을 사용하는 데에 있어 최종사용자가 어떤 라이센스를 적용받는지에 대한 것이다.

```
MODULE_LICENSE("GPL");
```

다음 세 줄의 명령은 모듈의 배포자와 설명 그리고 지원하는 디바이스 즉, 연결할 수 있는 하드웨어를 지정해주는 역할을 한다.

```
MODULE_AUTHOR(DRIVER_AUTHOR);
MODULE_DESCRIPTION(DRIVER_DESC);
MODULE_SUPPORTED_DEVICE("testdevice");
```

6.2 Netfilter 구현

Netfilter는 앞서 설명한 것처럼, 패킷이 통과하는 통로 사이에 커널 모듈을 넣어 패킷이 흘러가는 모든 것을 볼 수 있게 해주는 Framework이다. Netfilter는 다음 두 헤더 선언을 통해 사용될 수 있다.

```
#include <linux/netfilter.h>
#include <linux/netfilter_ipv4.h>
```

다음은 Netfilter를 커널 모듈로 Hook 하기 위한 일종의 main 함수로, Hook된 패킷이 아래 함수로 흘러 들어오게 된다. 모든 데이터는 sk_buff 구조체에 들어 있다.

```
unsigned long inet_aton(const char*);
unsigned int main_hook(unsigned int hooknum,
struct sk_buff *skb,
const struct net_device *in,
const struct net_device *out,
int (*okfn)(struct sk_buff*))
```

sk_buff 구조체의 데이터를 이용해 해당 데이터가 차단되어야 하는 데이터면 NF_DROP을 반환하고 통과해도 괜찮은 데이터면 NF_ACCEPT를 반환해 데이터 흐름을 조절할 수 있다.

```
return NF_DROP;
return NF_ACCEPT;
```

6.3 저수준 통신 IOCTL 구현

커널 모듈에서는 사용자 영역에 있는 프로세스와 통신을 할 수 없다. 즉 사용자가 명령을 직접 내릴 수 없는 것이다. 일반적인 C언어 환경이라면 프로세스간 통신에 Inter-process Communication 기능을

이용해 통신이 가능하지만 커널 모듈에서는 조금 다른 방법을 사용한다.

커널 모듈에서는 커널 모듈 자신을 가상 드라이버로 시스템에 등록하고, 사용자 영역의 애플리케이션은 이 가상드라이버를 통해 커널 모듈이 하드웨이인 것처럼 인식해 명령을 주고 받을 수 있게 된다. 이를 저수준 통신 IOCTL이라 한다.

다음 함수는 IOCTL 함수로, buf 변수에 데이터를 받아와 아이피나 포트를 차단목록에 등록하는 역할을 수행한다.

```
ssize_t drop_ioctl(struct file *filp, const char *buf, size_t count, loff_t *fpos)
{
        struct denystructure* pmsg = (struct denystructure*)buf;
        if(pmsg->cmd == ADD_IP)
        {
                char *addr = pmsg->addr;
                addip(addr);
        }
        else if(pmsg->cmd == ADD_PORT)
        {
                printk(KERN_ALERT "ADD PORT CALL");
                long port = pmsg->port;

                printk(KERN_ALERT "ADD PORT CALL %ld", port);
                addport(port);
        }
        return 0;
}
```

6.4 가상드라이버 구현

앞서 설명한 것처럼 IOCTL 통신을 위한 가상드라이버 등록은 device_create 함수를 통해 이루어지며, 디바이스 정보를 등록하는 alloc_chrdev_region, class_create, cdev_init, cdev_add 함수가 같이 사용된다.

```
int _ERR_;
_ERR_ = alloc_chrdev_region(&stDev, 0, 1, DEVICE_NAME);
if(_ERR_ < 0)
{
}

stClass = class_create(THIS_MODULE, DEVICE_NAME);
cdev_init(&stCDev, &fops);
_ERR_ = cdev_add(&stCDev, stDev, 1);

if(_ERR_ < 0)
{
}

device_create( stClass, NULL, stDev, NULL, DEVICE_NAME );
```

6.5 커널 파일시스템 제어 인터페이스 VFS(Virtual File System) 구현

커널 영역에서는 사용자 영역의 파일시스템에 접근을 할 수 없다. 이는 오버플로우 공격을 유발하며, 컴퓨터의 모든 기능을 자유롭게 접근제어 가능하며, 다양한 메모리에 직접 엑세스가 가능한 커널 모듈의 경우 오버플로우 공격은 매우 명적이게 된다. 따라서, 커널 모듈이 파일시스템에 엑세스하기 위해서는 VFS라는 기술을 사용해야 한다. VFS는 기존 파일시스템을 미러링해 가상의 파일시스템을 만들고 그 영역에 한정해서 커널 모듈 혹은 드라이버가 자유롭게 접근이 가능하도록 해주는 완충장치 역할을 맡는다.

다음 함수는 VFS를 이용해 파일시스템에 접근, 파일의 내용을 읽어 오는 것으로서, get_fs, set_fs는 기본 C언어 파일 입출력 함수이고, vfs_read 함수가 핵심이다.

file_read 함수의 반환은 int형식이지만, data매개변수에 읽어온 값을 넣어주게 되고, 외부 함수에서 이를 통해 읽은 데이터를 가져갈 수 있게 된다.

```
int file_read(struct file* file, unsigned long long offset, unsigned char* data, unsigned int size) {
  mm_segment_t oldfs;
  int ret;

  oldfs = get_fs();
  set_fs(get_ds());

  ret = vfs_read(file, data, size, &offset);

      set_fs(oldfs);
  return ret;
  }
```

다음 함수는 VFS를 통해 파일을 쓰는 함수이고, data매개변수를 파일에 쓰게 된다.

```
int file_write(struct file* file, unsigned long long offset, unsigned char* data, unsigned int size) {
  mm_segment_t oldfs;
  int ret;

  oldfs = get_fs();
  set_fs(get_ds());

  ret = vfs_write(file, data, size, &offset);

  set_fs(oldfs);
  return ret;
  }
```

6.6 sk_buff를 분석해 Protocol 분석기능 구현

앞서 설명한 Netfilter의 sk_buff에는 모든 패킷 데이터가 담겨 있다. 이때, iph에는 Internet Protocol의 데이터를 넣어주고 tcphdr에는 TCP 데이터를, udphdr에는 UDP 데이터를 넣어주게 된다.

그리고 다음 명령을 통해 saddr에 Source IP를, daddr에 Destination IP를, source에 Source Port를 dest에 Destination Port를 넣어주게 되고 이를 다른 기능에서 이용하게 된다.

```
  struct iphdr *iph = ip_hdr(skb);
  saddr = iph->saddr;
  daddr = iph->daddr;
  struct tcphdr *tcph = (struct tcphdr*)((char*) iph + sizeof(struct iphdr));
  struct udphdr *udph = (struct udphdr*)((char*) iph + sizeof(struct iphdr));
  source = htons(tcph ->source);
  dest = htons(tcph ->dest);
```

6.7 악성코드 샘플링 애플리케이션 구현

한 가지 더 필요한 것은, 악성코드를 어떻게 데이터베이스화 할 것이냐 이다. 악성코드 들은 스스로 변조시킬 수 있다 하지만, 리버스엔지니어링과 디지털포렌식 기술을 바탕으로 보면, 지울 수 없는 고유한 흔적들이 있다. 애플리케이션 시그니쳐 혹은 애플리케이션 핑거프린트라고도 하는 이것은, 개발된 환경, 실행될 환경 등을 담고 있는 중요한 내용인데, 디지털 범죄자들을 잡아내는데 유용하게 쓰이고 있는 정보이다.

아래 소스는 파일의 앞 1024바이트를 가져와 db에 저장하는 애플리케이션으로 파일의 앞 1024바이트는 실행파일인지 아닌지, 리눅스 실행파일인지 윈도우 실행파일인지 등을 알려주는 정보를 갖고 있다.

이 외에도 <별첨 1>을 보면, 파일의 여러 가지

시그니쳐를 분석해 해당 정보를 데이터베이스화 하면 악성코드 엔진이 만들어지게 된다.

```
if(fp = fopen("/home/david/Downloads/file.exe", "rb"))
{
        if(fp2 = fopen("/home/david/Desktop/dev2/database.db", "wb"))
        {
        str = malloc(buff_size+5);
        fgets(str, buff_size, fp);
        fputs(str, fp2);
        fputs(",", fp2);
         fclose(fp);
        free(str);
        }
}
```

7. 결과

7.1 Google App Engine

7.1.1 구글에 등록된 프로젝트

PROJECT NAME	PROJECT ID
linux filtering	linuxfiltering

7.1.2 Google App Engine Request Log

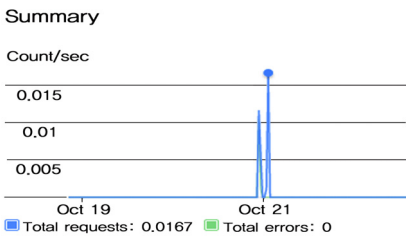

그림 10. 구글앱 엔진 Request Log
Fig. 10. Google App Engine Request Log

7.1.3 Google App Engine 버전관리

VERSION	SIZE	RUNTIME	INSTANCES	DEPLOYED
1 (default)	3.7 KB	php	0	4 days ago

7.2 Kernel Module Install

7.2.1 Kernel Module Install Command

```
~/Desktop/dev2$ sudo insmod drop.ko
```

7.2.2 Kernel Module System Log

```
kernel: [37466.094420] Module Install Complete
kernel: [37466.094426] VFS I/O Test
kernel: [37466.094446] DATA:.class,
kernel: [37466.094446]
kernel: [37466.152419] VFS I/O Test Complete
```

7.3 파일 다운로드

7.3.1 데이터베이스에 등록된 윈도우기반 실행 파일 다운로드 시도 시 보여지는 해당 파일의 정보

다운로드 했으나 크기가 0bytes인 것을 알 수 있다.

8. 결론 및 향후 연구 과제

리눅스는 오래전부터 이미 상용 방화벽 장비에 뒤지지 않는 성능을 발휘하고 있다. 또한 최근 방화벽 기술의 경향은 네트워크 트래픽 필터링과 침입 탐지 분야의 계층별 컨텐츠 검사 기술의 결합인데, 이를 Netfilter Framework를 이용해 소프트웨어적으로 구현한 사례는 없었다. 구현 방식도 까다롭고 알려진 정보도 미흡할뿐더러, 스마트폰 등 다양한 IT기기의 등

장으로 응용소프트웨어로 개발자들이 몰리면서, 저수준 소프트웨어를 구현할 수 있는 프로그래머가 줄고 있는 탓이다. 여기까지 개발한 연구과제를 토대로 리눅스 기반의 강력한 기술들을 더 융합하게 된다면, 고성능 네트워크 장비를 뛰어 넘는 시스템을 구축할 수 있으리라 여겨진다. 향후 연구과제로는 이 기술과 안드로이드 운영체제간의 결합 방안을 모색해보고, 아울러 셀룰러 네트워크의 공격과 방어에 활용할 수 있는 기술을 구현해보고자 한다.

참 고 문 헌

〔국내 문헌〕

[1] 안지용 (2001), “고속 패킷 필터링 알고리즘 개발”, 숭실대학교 정보과학대학원 석사 학위논문.

〔웹사이트〕

[2] 경희사이버대학교 미래고등교육연구소 (2012), “심층 패킷 분석은 인터넷을 어떻게 염탐하는 것일까”, 학술웹진 칼럼(http://igcs.khcu.ac.kr/board/list.jsp?m=50026&SITE_GRP=IGCS).

[3] http://www.tldp.org/LDP/lkmpg/2.6/html/(The Linux Documentation Project Official Site).

[4] http://en.wikipedia.org/wiki/Loadable_kernel_module (Wikipedia US, Information of Kernel Module).

[5] https://www.kernel.org/(The Linux Kernel Archives).

박 대 승 (Daeseung Park)

남서울대학교 컴퓨터학과 공학사를 취득하였다. 관심분야는 소프트웨어 개발이다.

김 수 민 (Soomin Kim)

남서울대학교 컴퓨터학과 공학사를 취득하였다. 관심분야는 소프트웨어 개발이다.

유 한 섭 (Hanseob Yoo)

남서울대학교 컴퓨터학과 공학사를 취득하였다. 관심분야는 소프트웨어 개발이다.

문 송 철 (Songchul Moon)

KAIST에서 MIS전공으로 공학석사학위를 취득하였고 국민대학교에서 MIS전공으로 정보관리학 박사학위를 취득하고 현재 남서울대학교 컴퓨터학과 교수로 재직 중이다. 한보정보통신(주)에서 철강SI사업부장, 관리이사와 가나시스텍(주) 사장으로 재직하였다. 정보시스템감리인 자격을 보유하고 있으며 주요 논문은 정보처리학회지, 한국IT서비스학회지, 디지털콘텐츠학회지, ICCMSE 등의 국내외 학술지와 한국IT서비스학회, 경영정보학회, ICCMSE 등의 국내외 학술대회에서 논문을 발표, 게재하였다. 주요 관심분야는 소프트웨어공학, 시스템 분석 및 설계, 정보시스템 감리이다.

Development of Kernel based High Speed Packet Filtering Imbedded Gateway and Firewall Using Cloud Database

Daeseung Park* · Soomin Kim* · Hanseob Yoo* · Songchul Moon**

ABSTRACT

This paper develop curnel based high speed packet filtering imbedded gateway and firewall using cloud database. This study develop equipment include of predict function through bigdata analysis using cloud system. This equipment include intrusion prevention for network attack, and include system security function of L7 switch based contents. This study can improve security level of little company and general family. This study can pioneer a new market. This study can develop high perfomance switch and replacement of existing security equipment. This study proposed new next generation algorithm for constuction of high performance system from low specifications.

Keywords: High Speed Packet Filteringsmart, Cloud Database, Security Level, High Perfomance Switch, Gateway, Firewall.

* Namseoul University, Department of Computer Science, undergraduate
** Corresponding Author, Namseoul University, Department of Computer Science, Professor, moon@nsu.ac.kr